StripStream

a comic book reader format for the Commodore 64

User manual

The information in this manual refers to version 1.1 of the editor and player software

Introduction

This manual holds information for anyone who wants to use the StripStream editor program to create StripStream releases (digital comic books for the Commodore 64 computer) in the form of a .TAP file or real tapes. StripStream is a digital comic book format to be used in combination with audio-cassette based storage devices like the 1530 and 1531 on a C64. This manual also holds all technical information regarding data transfer of raw data and high level packages used by the StripStream protocol.

A comic book format for the C64 is fun, but how can I demonstrate this when I can't even draw a decent stick figure myself. The challenge with the C64 are it's limitations, resolution is low, color depth and memory are highly limited. Therefore, for the demonstration of the StripStream format high contrast images are to be preferred. And from a practical point of view, those modern text balloons take up too much room that's better to be spend on images.

I decided to use the "Kapitein Rob" book: De avonturen van het zeilschip "De Vrijheid" (Captain Rob, The adventures of sailingship "The Freedom"). It's a Dutch book which started as a comic strip published in the Dutch newspaper "Het Parool" on December 11, 1945. These daily strips were bundled into a small booklets. Of which more than 70 different adventures were released.

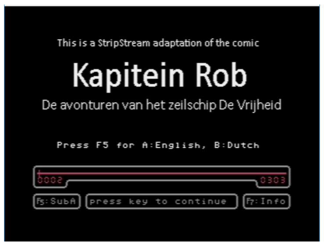
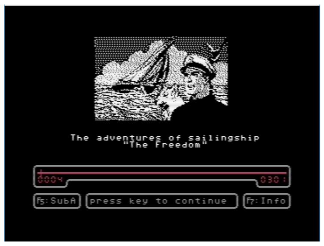
This iconic comic was a small part of my fathers childhood, he has really seen it in the newspapers and even bought some of these small booklets from the newspaper/cigar stores at that time, which unfortunately, he no longer has. But whenever we talk about comic books at home, "Kapitein Rob" is mentioned. So you can imagine that this book jumps up in my mind whenever I think about early Dutch comics. Although, it wasn't until the StripStream project that I really started to appreciate the beautiful black and white images of this series. And unlike "modern" comics there are no text balloons for displaying the text. Instead the text is displayed underneath the images. Thinking about the fact that the C64 is a computer from my own youth and the "Kapitein Rob" book is a comic from the youth of my father, I thought it was a nice idea to combine both. By using the first book in this series as an example for my project I created an homage to the iconic Dutch comic but also to my father, who was just a child that liked a good story.

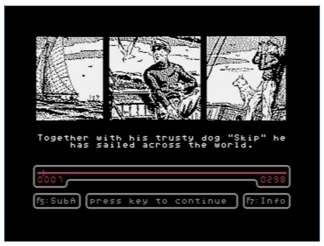
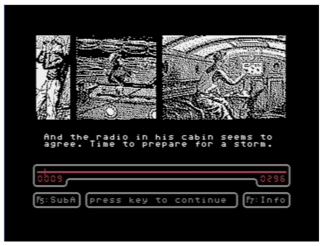
De avonturen van het zeilschip „De Vrijheid"

Hier zijn de drie hoofdpersonen van het verhaal, dat komen gaat. Het merkwaardig jacht „De Vrijheid", zo'n schuit, waarvoor een man heel wat over heeft om haar te bezitten. Kapitein Rob van Stoerum is zo'n man. Hij is, als vele zeelieden, zwijgzaam en onverstoorbaar en hij heeft van die ogen, die altijd in de verte schijnen te kijken. Hij is de kalmte in eigen persoon en is er zo een, die alle moeilijkheden overwint. En dat moet ook, want het is gek, waar hij ook vaart met zijn schip, altijd beleeft hij spannende en vreemde avonturen. Helemaal alleen is hij nooit, want zijn hond, Skip, vaart altijd met hem mee en Skip heeft meer van de wereld gezien dan jullie allemaal bij elkaar. Hij voelt zich aan boord beter thuis dan aan de wal; hij heeft vier echte zeepoten gekregen en al springt het buiswater nog zo hoog over hem heen, al ligt de schuit zo scheef, dat ze water schept, Skip voelt er zich lekker bij.

However, we must realize that the Commodore 64 is just a home computer of the 80's, it isn't a really powerful computer, memory and graphics are highly limited. Therefore the StripStream adaptation of this comic book doesn't look as pretty on the C64's monitor as it does on real paper. Also the text of the story had to be compressed significantly in order to fit the project, but the 300 images of the book tell the real story very well and therefore the main focus of the project was to include 'm all into a single StripStream release.

Therefore the StripStream release of this book is not to be compared with the real version of the book, the real book simply looks much better and has much more text. So after reading the StripStream version, you really should also read the actual book.


This is a StripStream adaptation of the comic
Kapitein Rob
De avonturen van het zeilschip De Vrijheid
Press F5 for A:English, B:Dutch
0002
0303
F5:SubA
press key to continue
F7:Info



Together with his trusty dog "Skip" he has sailed across the world.
0007
0298
F5:SubA
press key to continue
F7:Info

And the radio in his cabin seems to agree. Time to prepare for a storm.
0009
0296
F5:SubA
press key to continue
F7:Info

Mostly for the fun of it, but also to actually have something physical to show at meetings, other than a .EXE and .TAP file on a website, I created a nice label and a j-card for my own personal copy of the tape. This simply allows me to present my project with just that little bit of extra flair. After all a real tape is something you can touch and experience, which is what retro computing is all about.

The StripStream release, based on the "Kapitein Rob" series, is only an example, a showcase for the StripStream project to demonstrate that digital comic books would have been possible in the 80's. The fact that I've used a modern PC to achieve this goal, is just a minor detail. The end result shows that a C64 in combination with audio-cassette technology for storage is perfectly suited to perform the task of reading digital comic books in a convenient way. I'm pleased to say that the rightful owners of the Kapitein Rob series have granted me special permission to use the first book of this series as an example for my project. As mentioned earlier, I really want to emphasize the technical limitations of StripStream, prevent the book from being showed in it's true glory. The .TAP file of my example project may not be used for commercial purposes and may only be published on: https://janderogee.com

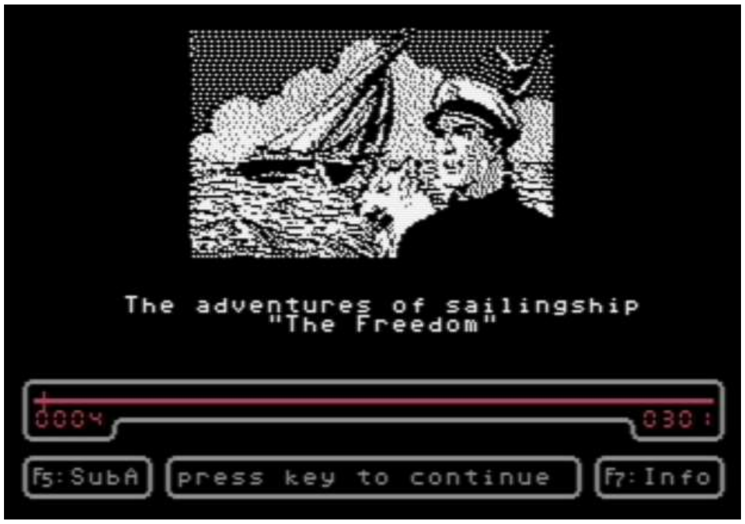
Jan Derogee, 2023

Table of Contents

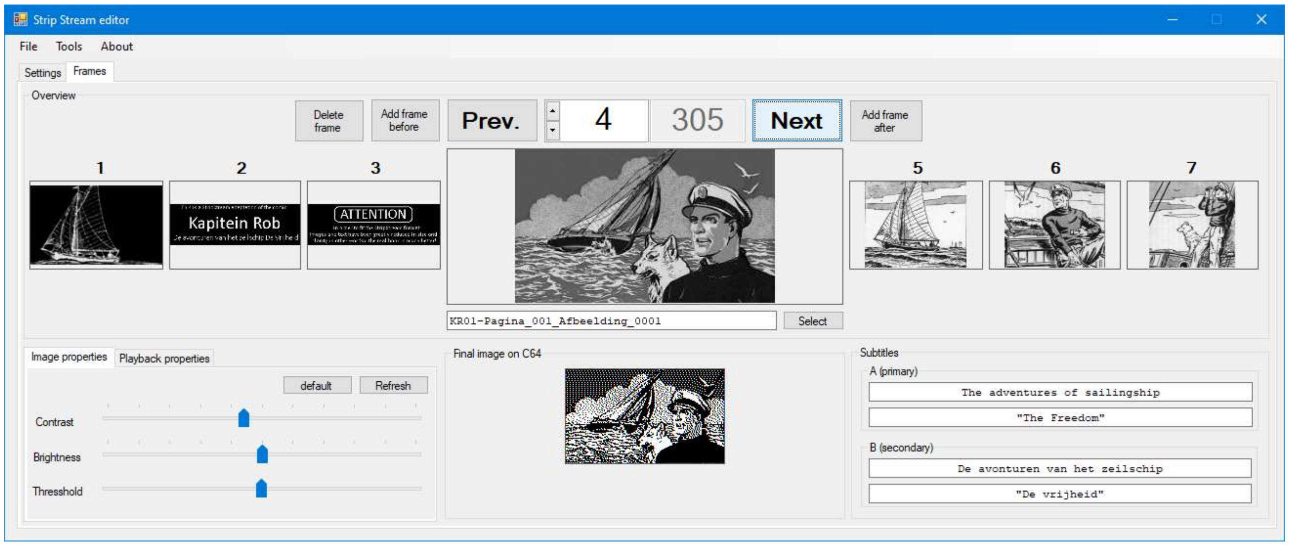
1 What is StripStream

StripStream is a protocol that allows for comics books to be loaded and displayed directly from tape on a Commodore 64. The Commodore 64 is a computer hat was released 1982 and remians popular even 40 years after its release. The StripStream protocol was released in 2023, just 3 or 4 decades too late to be really effective. But it might provide lots of fun for the retro computing community as it allows to use the 1530 and 1531 datasette for something other than loading games.

Below is a screenshot of how the StripStream player looks like on a C64 and a screenshot of the "Frames" section of the Strip Steam editor program.



The adventures of sailingship
"The Freedom"



KR01-Pagina_001_Afbeelding_0001

A (primary)

The adventures of sailingship
"The Freedom"

B (secondary)

De avonturen van het zeilschip
"De vrijheid"

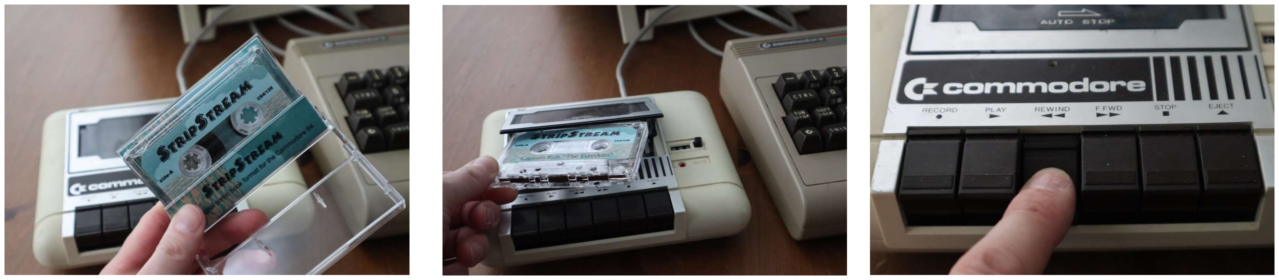
Since this protocol is designed with the purpose of displaying and scrolling of B/W HIRES images from tape, loaded by a C64 using a 1530 or 1531 datasette (or compatible system), there are some important limitations to keep in mind. StripStream is able to store hundreds of images, with accompanying subtitles in two languages including the playback software it all fits onto a single tape. Would this have existed in the 80's, it truly could have been revolutionary.

2 StripStream usage

The idea behind StripStream is to playback comic books on a real system as it was designed in the 80's. No fancy modern stuff, just a Commodore 64 with monitor and a datasette with a StripStream tape. Unfortunately, not everyone is able to get their hands on such equipment but in these modern times thee are plenty of Commodore 64 emulators around. Perhaps one of the best is named VICE however an emulator cannot accepts real tapes, so you'll have to use an emulated tape instead, emulated tapes come in the form of a file ending with the extension .TAP, which is a format that well established in the Commodore retro computing scene and does it's job perfectly.

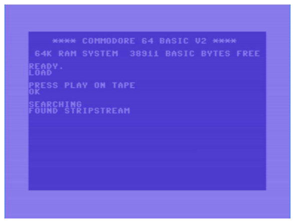
2.1 Using a real tape on real hardware

In order to read a StripStream tape, simply insert the tape (side-A) into your datasette and rewind the tape to the beginning. You may reset the tape counter to 000, because it's fun.

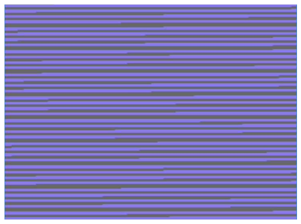


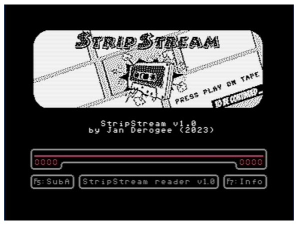
Type LOAD <return> on your Commodore 64.

When the computer states "Press PLAY ON TAPE" press the play button on the tape. Now lean back and wait until the program loads, according to modern standards, this will take pretty long. According to 80's standards, it loads amazingly fast, just 40 seconds.



```
    **** COMMODORE 64 BASIC V2 ****
 64K RAM SYSTEM  38911 BASIC BYTES FREE
READY.
LOAD
PRESS PLAY ON TAPE
OK
SEARCHING
FOUND STRIPSTREAM
```

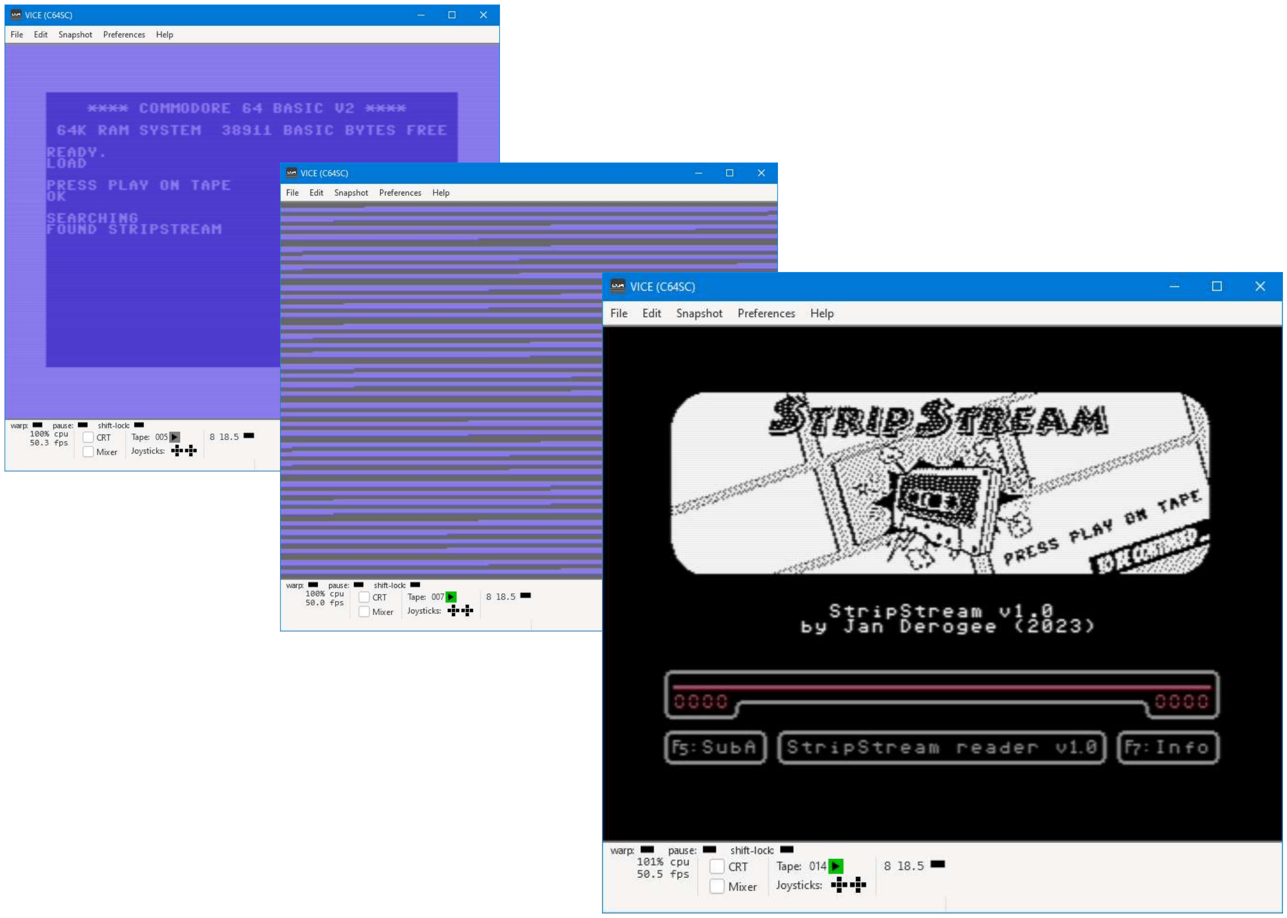




Follow the instructions on the screen and enjoy the show.

2.2 Using a .TAP file on an emulator

Reading a StripStream .TAP file on an emulator can be done in various ways. The very versatile and accurate emulator VICE is perhaps the best and most used emulator in the Commodore retro computing scene. It can be downloaded from: https://vice-emu.sourceforge.io/



```
    **** COMMODORE 64 BASIC V2 ****
 64K RAM SYSTEM  38911 BASIC BYTES FREE
READY.
LOAD
PRESS PLAY ON TAPE
OK
SEARCHING
FOUND STRIPSTREAM
```

```
StripStream v1.0
by Jan Derogee (2023)
0000                                0000
```

Method 1
This is the official way, start VICE.
Go to: FILE → attach datasette image → select file ending in .TAP → click attach/load
Type: LOAD <return>
Go to: FILE → datasette controls → start
Follow the instructions on the screen and enjoy the show.

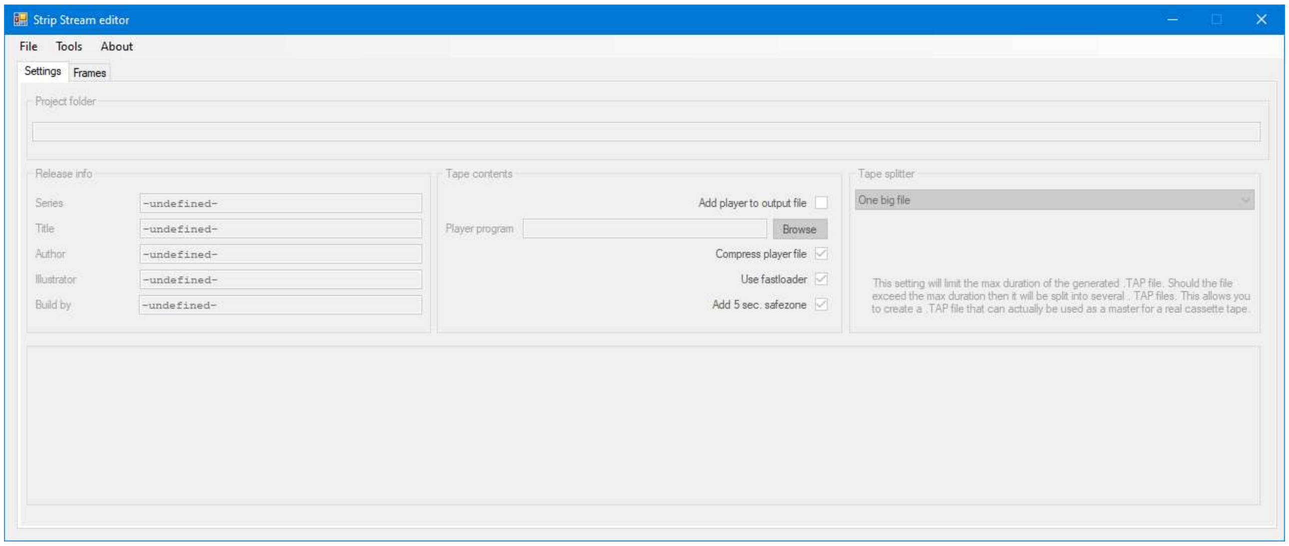
Method 2
The preferred method would be to make a shortcut to the VICE emulator on your desktop
Drag the .TAP file of interest onto the shortcut and drop it there.
VICE will now automatically attach the tape image, type LOAD for you and press play.
Follow the instructions on the screen and enjoy the show.

3 StripStream editor

Reading StripStream releases is fun, but making your own StripStream releases might be even more fun. The editor tool allows you to create your own .TAP file from which you can make a real tape.

3.1 Creating a project

When the StripStream editor program is started, everything is disabled and there is not much you can do. That is because when the program starts, it first need to open (or create) a project.



Project folder

Release info

Series	-undefined-
Title	-undefined-
Author	-undefined-
Illustrator	-undefined-
Build by	-undefined-

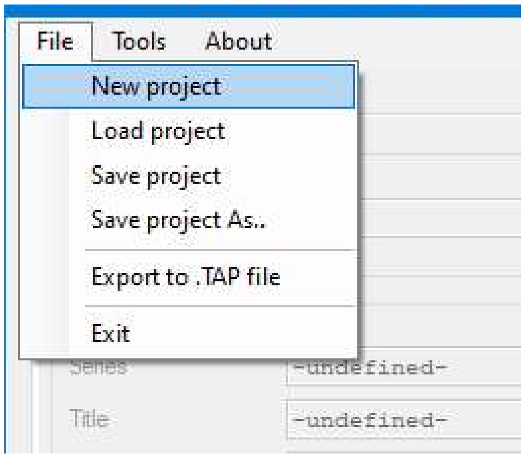
Tape contents

Add player to output file ☐
Player program
Compress player file ✓
Use fastloader ✓
Add 5 sec. safezone ✓

Tape splitter

One big file

This setting will limit the max duration of the generated .TAP file. Should the file exceed the max duration then it will be split into several .TAP files. This allows you to create a .TAP file that can actually be used as a master for a real cassette tape.



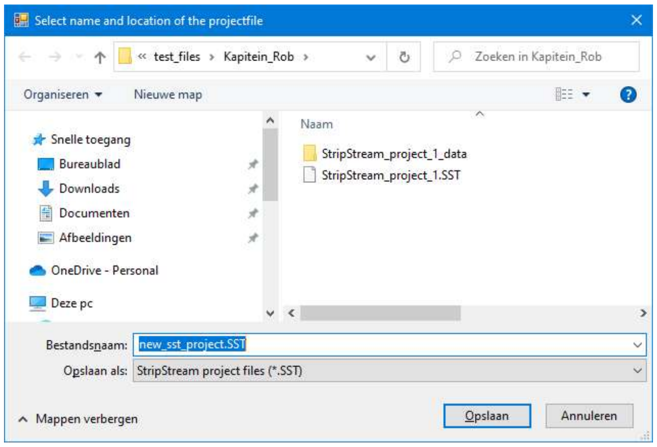
Despite the fact that making a StripStream release is a project in itself, a StripStream project consists of many lines of text and many images and all require a lot of settings. All that information needs to be stored somehow and somewhere. In order to store all this data in a convenient way a project file along with a project folder is required. The project folder will be holding all the images as used in the StripStream release and is automatically created upon entering the filename and location of the projectfile.

Go to:

File → New project

This will result in a file requester that ask for the name of the project file.

The suggested project filename for a new file is new_sst_project.SST, but you can change that into any name you'd like.


Select name and location of the projectfile

« test_files › Kapitein_Rob ›

Zoeken in Kapitein_Rob

Organiseren ▾ Nieuwe map

Snelle toegang
Bureaublad
Downloads
Documenten
Afbeeldingen
OneDrive - Personal
Deze pc

Naam
StripStream_project_1_data
StripStream_project_1.SST

Bestandsnaam: new_sst_project.SST
Opslaan als: StripStream project files (*.SST)

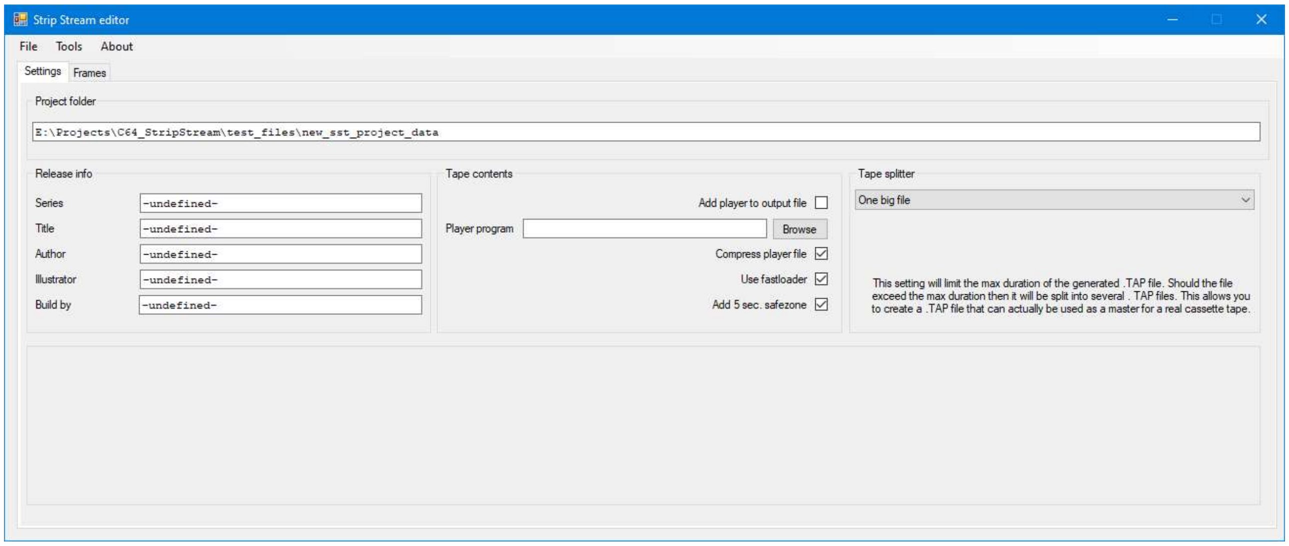
Mappen verbergen
Opslaan
Annuleren

If you would enter StripStream_project.SST, a file of such a name will be generated and it will be holding all the project settings. The project images are saved to the project folder folder, that folder is created automatically (and in the same folder as the project file) and will be named just like the project file, but with the addition of _data.

So after making a project, you will see one file ending with .SST (the project file) and one folder ending with _data (the project folder). These two items need to be together at any time. Similarly like are like: Laurel and Hardy, Gumball and Darwin, Bassie en Adriaan. In other words, if these files are not together, things simply will not work. So keep this in mind should you decide to relocate your project. After creating the project file and folder, the program allows you to configure the project. Regarding the settings panel, there are mainly 3 sections release info, Tape contents and Tape splitter.

Release info

Here you are required to enter the meta data of the project. These fields are a bit like the first pages of an ordinary book, holding important credit info.


Strip Stream editor

File Tools About

Settings Frames

Project folder

E:\Projects\C64_StripStream\test_files\new_sst_project_data

Release info

Series -undefined-
Title -undefined-
Author -undefined-
Illustrator -undefined-
Build by -undefined-

Tape contents

Add player to output file ☐
Player program [ ] Browse
Compress player file ☑
Use fastloader ☑
Add 5 sec. safezone ☑

Tape splitter

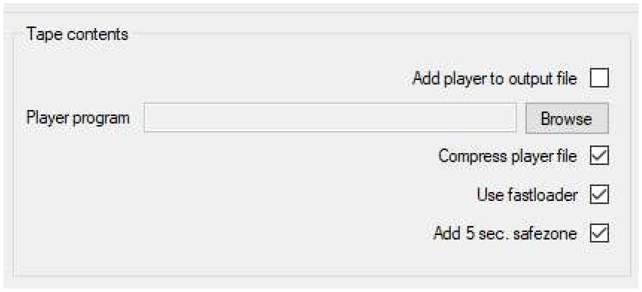
One big file

This setting will limit the max duration of the generated .TAP file. Should the file exceed the max duration then it will be split into several . TAP files. This allows you to create a .TAP file that can actually be used as a master for a real cassette tape.

Series	: name of the main character of the book
Title	: title of the book (i.o.w. the name of the adventure of the main character)
Author	: name of the person who actually wrote the story
Illustrator	: name of the person who actually made the drawings
Build by	: initiator of the StripStream release (enter your own name here)

Tape contents

Since StripStream is a tape based format, the release is referred to as a tape (or .TAP file). On that tape can be much more than just the story and pictures. The player program for example. Because a StripStream release is useless without a player to play the data with. By including the player on the tape, you can always be sure that the data can be played back. When the player would be on a disk and the story on tape, well, it's a matter of time before they get separated and both become useless.


Tape contents

Add player to output file ☐

Player program [Browse]

Compress player file ☑

Use fastloader ☑

Add 5 sec. safezone ☑

Add player to output file: Normally this box should be checked, because you really want to add a player to the beginning of the tape. However in some very specific situations you may not want to add the player, I cannot give you an example, but should the situation arise, then it's a nice to know you have that option.

Player program: Normally you'd have selected the StripStream player program selected here (program.prg). But if you have written your own player or some other program that you may want to add to the beginning of the tape, then you can select it here.

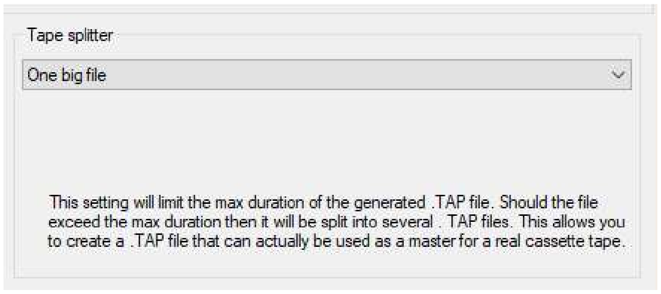
Compress player file: Normally this box should be checked. The player program is a few kB, which may take a while to load, by compressing the file to a smaller size, loading time may be decreased significantly. When the program has been loaded it will decompress to its normal size, this take only a few seconds. Compression is done using the tool named "Exomizer".

Use fastloader: Normally this box should be checked. The player program is a few kB even compressed loading with the normal Commodore kernal loader will require a few minutes. This is a waste of time and tape but by using a fastloader (Turbo tape) the loading time is decreased significantly. This option may be disabled to improve the odd of loading this program succesfully, however, if you cannot load a program with this fastloader enabled, you may have problems with the playback of the tape in general, meaning that you might not also be able to play the StripStream data properly. In other words, there is no good reason to disable this option.

Add 5 sec. safezone: By checking this box, there is a 5 second gap of silence in between the player program and the StripStream data. In many cases this is not needed, however if it takes along time to decompress the loaded program and if the tape is still running, then data might be lost. By adding a 5 second gap there is no data to loose. This is just one single 5 second gap of white space, directly after the player program at the beginning of the tape. Checking this box also ensures that the splash screen of the player program is fully visible for a few extra seconds (because for the first few seconds on the tape, there is no data to load), which may be a desired effect.

Tape splitter

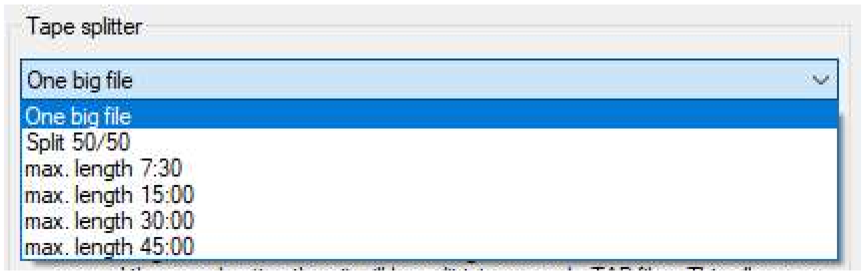
StripStream is all about tape, therefore it should be possible to make a .TAP file that is able to be transferred to a real tape. It would be pretty lame if you created a file that would not fit onto a real tape. Therefore there's the tape splitter. After the creation of the .TAP file, the splitter routine analyzes the size of the created .TAP file and decides if it will fit onto the size of the tape as selected in the dropdown box. If you don't want the splitter to do any splitting, just select "One big file" as shown in the image below. This is perfectly fine for anything emulator based. But in real life, this isn't always what you need.


Tape splitter

One big file

This setting will limit the max duration of the generated .TAP file. Should the file exceed the max duration then it will be split into several . TAP files. This allows you to create a .TAP file that can actually be used as a master for a real cassette tape.

In real life situations, you really need to select another setting, below a screenshot of the options. The splitter splits at the appropriate moments, just imagine what would happen if it didn't, then the user might be forced to flip the tape in the middle of loading image data and that would corrupt the image. Therefore the splitter always splits the data before the start of a new frame. The created "split" files are additional, so you might end up with 3 files. The one big file and the two additional split files. This way you have the "one big file" that is perfect for the emulator and the smaller files that are perfect for writing to a real tape.

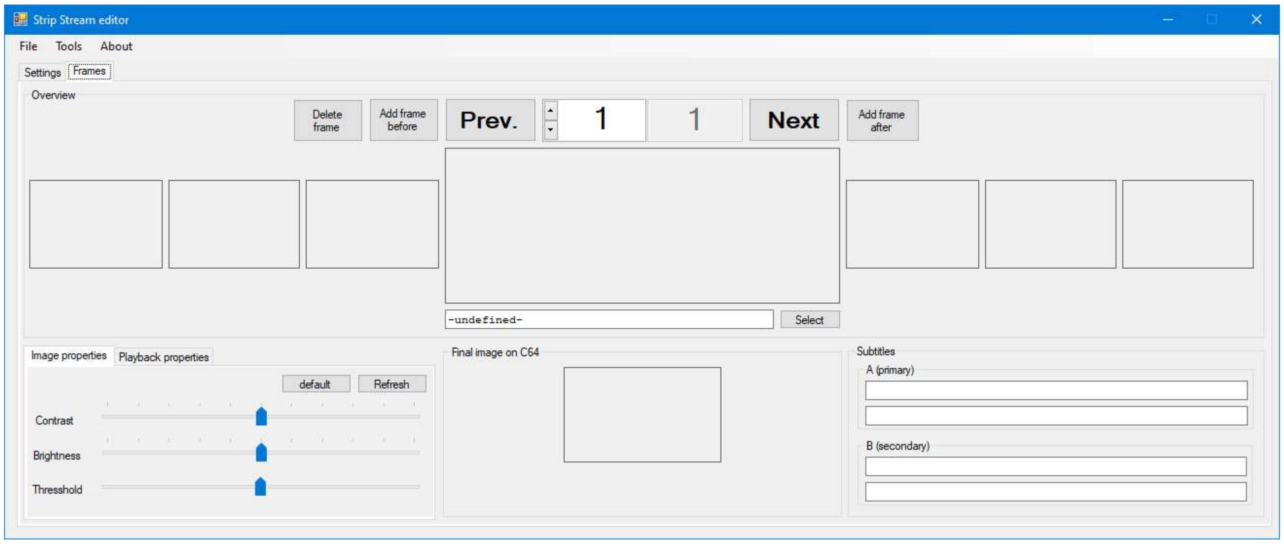


One big file:	splitter functionality is disabled
Split 50/50:	splits .TAP file into two even parts (useful for custom length tapes)
Max. length 7:30:	splits .TAP file before reaching the 7:30 mark (useful for C15 tapes)
Max. length 15:00:	splits .TAP file before reaching the 15:00 mark (useful for C30 tapes)
Max. length 30:00:	splits .TAP file before reaching the 30:00 mark (useful for C60 tapes)
Max. length 45:00:	splits .TAP file before reaching the 45:00 mark (useful for C90 tapes). Please keep in mind that the use of C90 tapes is to be avoided, due to the fragility of this type of tape, which is much thinner in order to fit into the cassette. This option was added merely for completion of the list of available tape sizes.

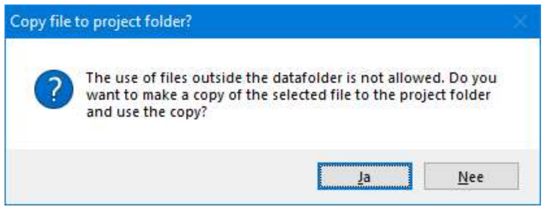
3.2 Creating the story

This is what's it is all about, the story, the collection of images and text. In order to add some text and pictures, you'll need to go to the "Frames" section of the program.

3.2.1 Adding images



When you start a new program, everything is still empty, there is already a first frame. To which you can add your picture. In the center of the screen is a large window that holds the current image and next to that are the images before and after, this allows you to get a clear view of the order of the images in the story. Underneath the current image is the filename with a browse button, press the browse button to load your image. A file requester will appear and you can select the image you want to use. Unless this image is already in the project data folder, the program will ask you to copy it there. You should answer with "Yes" in order to copy the file to the project data folder. A file located anywhere else simply cannot be used by the StripStream editor program. This because the editor needs all the files to be together in the same place.

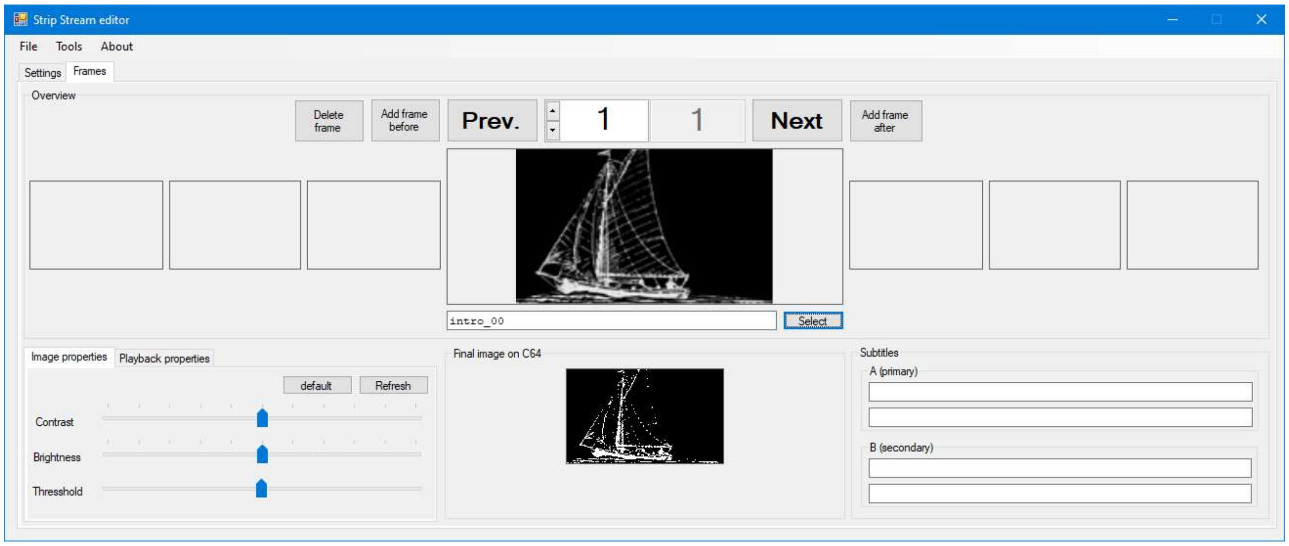


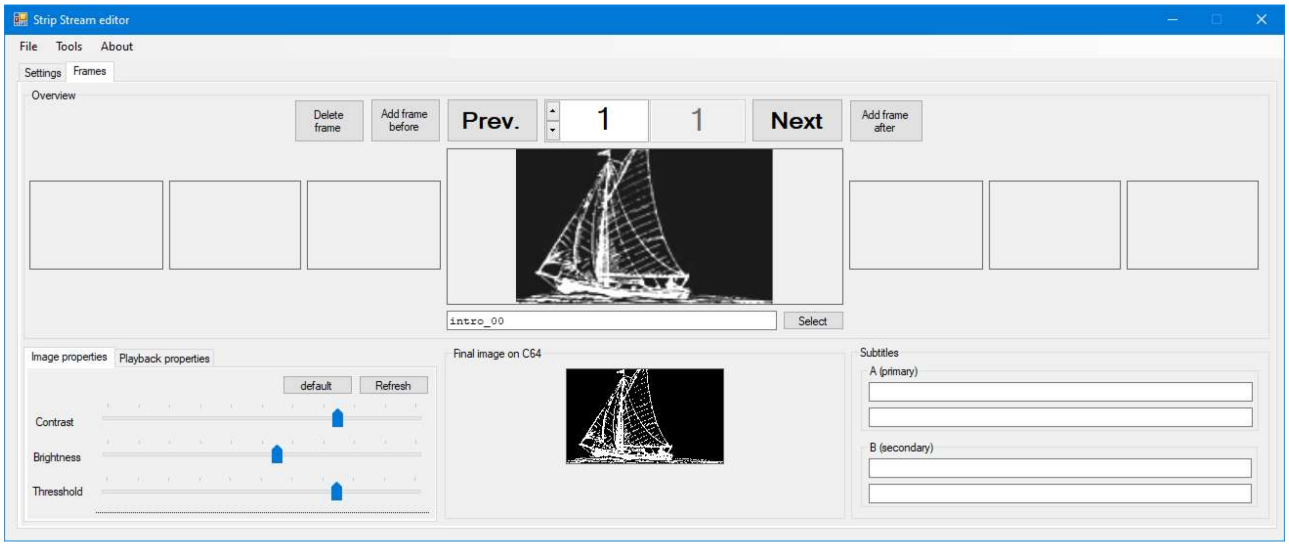
The use of files outside the datafolder is not allowed. Do you want to make a copy of the selected file to the project folder and use the copy?

Tip:

In order to get the best results, it helps to crop the images. Otherwise uninteresting parts in the image consume pixels that are better used for the parts of the picture that are important. Therefore crop the image to the absolute minimum size of things in the image you want to show. This is mainly for the vertical size of the image, since this is a fixed size, the horizontal size is much more forgiving as larger images are allowed to keep on scrolling, which in some situations might even be a desirable effect.

When the image has been selected and copied to the project data folder, you will see it in the editor. And you will notice that it doesn't look optimal. That's because StripStream only supports pure B/W images. So you need to play with the contrast, brightness and thresshold settings in order to get a properly looking dithered image. These settings are located in the section "Image properties". The image properties, allow you to select how the image looks, these are settings that must be done for each image separately.





When the image looks optimal, you can change the playback properties. This is located in the section "Playback properties".

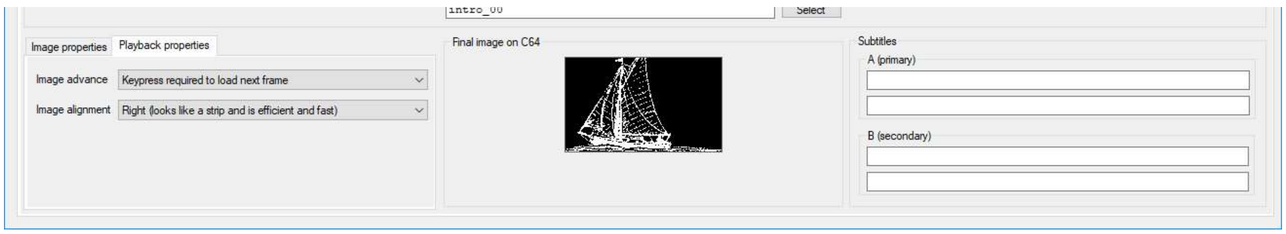


Image advance: Keypress required to load next frame

Image alignment: Right (looks like a strip and is efficient and fast)

The playback properties, allow you to select how the image is presented. These are settings that must be done for each image separately.

There is the “Image advance” setting.

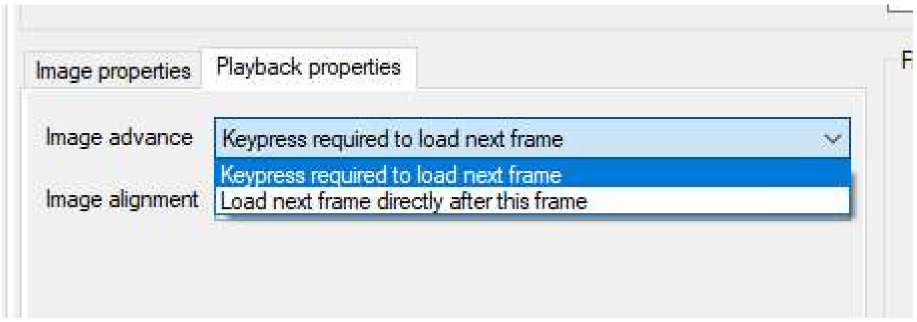


Image advance: Keypress required to load next frame

- Keypress required to load next frame
- Load next frame directly after this frame

Keypress required to load next frame: when this option is selected, the tape will be stopped after the loading of the image, then the user is requested to press a key in order to continue. While the tape is stopped the text of the currently visible subtitles are send to the userport as RS-232 (TTL) signals. This allows a speech synthesizer, like SSSSAM or the Votrax type ‘n talk, to speak the text (don’t expect to much of it, most subtitles will not be optimized for speech). Because this option forces to the tape to be stopped, it will add a 1 second gap on the tape after the image, this in order to facilitate the tape mechanism to slow down and speed up without any data loss. When this mode is active, the image shown on screen is allowed to be printed to a real printer (see section printer support)

Load next frame directly after this frame: when this option is selected, the tape will continue to play and the next image is loaded automatically. However, readabillity is not improved and the subtitle text will not be send to the userport. This option is therefore merely to allow for a simple scroll effect of an image, for example during the start of the tape, simulating a boat to sail by or plane with a large text banner to fly from right to left. When this mode is active, the image shown on screen is allowed to be printed. When this mode is active it is not possible to print the image to paper (unless you use a freezer cartridge, but that’s without the scope of this document).

There is the "Image alignment" setting.

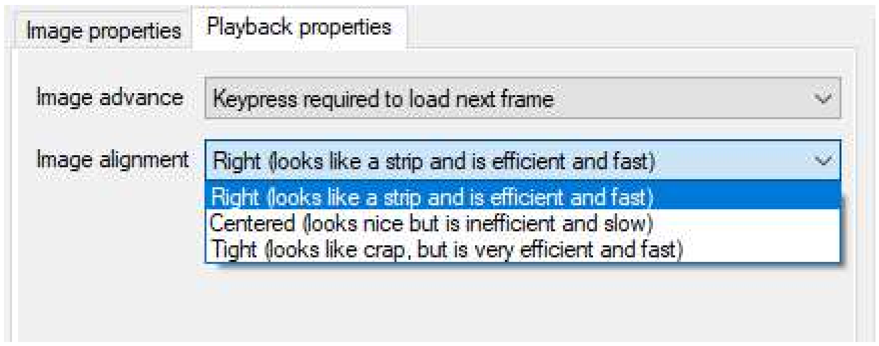
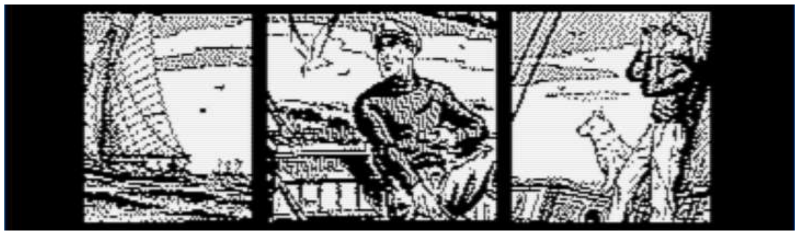


Image advance: Keypress required to load next frame

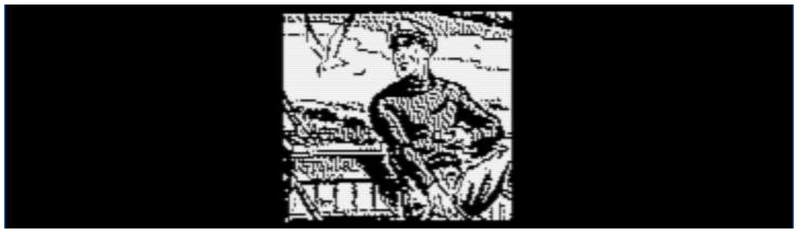
Image alignment: Right (looks like a strip and is efficient and fast)

- Right (looks like a strip and is efficient and fast)
- Centered (looks nice but is inefficient and slow)
- Tight (looks like crap, but is very efficient and fast)

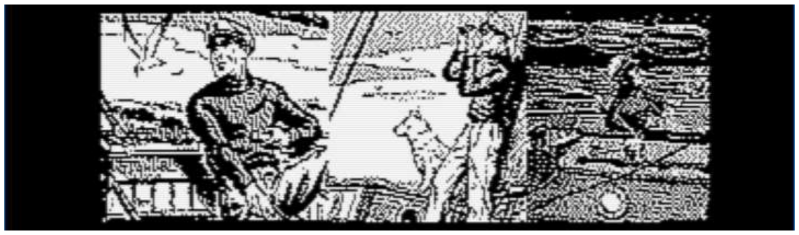
Right: when this option is selected for all images (it is the default setting) the images will scroll from right to left on the screen and give the appearance of a large filmstrip. Each image is separated from the next image by a series of 8 vertical black lines (on the right side of the image), resulting in a thick black separator line between the current image and the next.



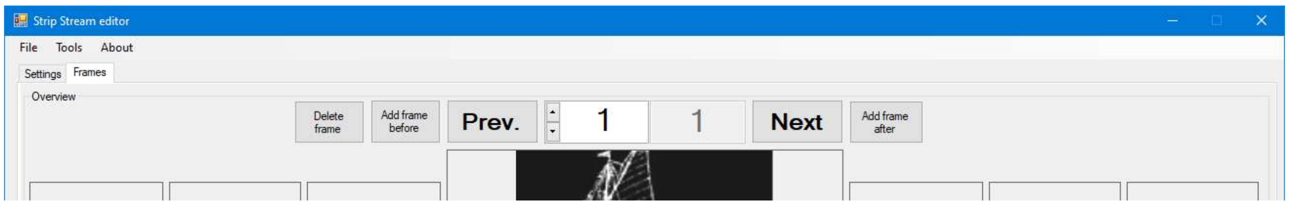
Centered: when this option is selected, the image will be centered during loading by the use of additional black lines on the left and right side of the image. This means that, when completely loaded, only this image will be visible. Unfortunately, centering the image requires the use of many extra black lines, information that also needs to be stored onto the tape, consuming precious tape space and increasing the loading time per image.



Tight: when this option is selected, there will be no space in between the current image and the next, meaning that both images will be joined seamlessly. Although this does save storage space and loading time, making it the most efficient option, it also looks like crap. It's a bit confusing and it doesn't have that iconic comic book look.



Adding more images is just a matter of clicking on the buttons “Add frame before” or “Add frame after”, which will add a frame before or after the current frame.



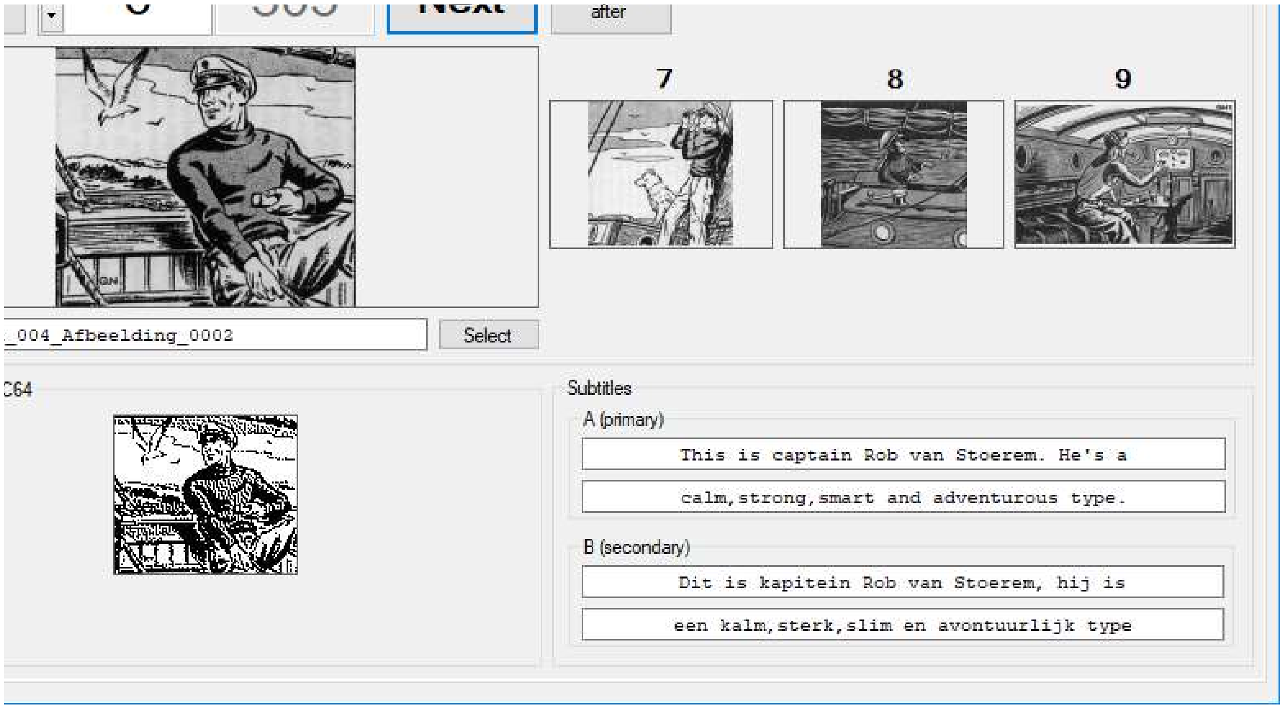
Moving a frame is not possible, so if you want to move a frame, you can’t. Which means that you need to delete the frame where you don’t need it and recreate it where you do.

If you want to navigate through the frames, click on the “Prev” or “Next” button, or simply click on the smaller images to jump that their corresponding frame. You can also change the number on the frame indicator, by clicking on the little up-down arrows, but the most convenient is perhaps the ability to enter the direct frame number into the frame indicator.

3.2.2 Subtitles

Subtitles are used as a means of adding text to the images. Because text balloons simply eat up too much hires screen space, room that could be better spend on the drawing. There is space for two lines of subtitles per frame. And in two languages. This way you can create a StripStream release for a broader audience. For instance the wonderful story of Kapitein Rob, the book used for demonstrating the power of StripStream is a Dutch comic, but this makes it impossible to read for anyone who does not speak Dutch. But by adding room in the protocol to support two languages, and making the second language English, suddenly much more people can enjoy this release.

Unfortunately, two lines of text per frame isn't very much, this is a compromise that has been made in order to fit everything onto the screen and the tape, since 1980's technology does have it's limitations.



_004_Afbeelding_0002

A (primary)

This is captain Rob van Stoerem. He's a

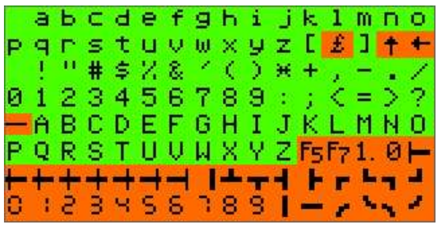
calm,strong,smart and adventurous type.

B (secondary)

Dit is kapitein Rob van Stoerem, hij is

een kalm,sterk,slim en avontuurlijk type

The subtitles are two sets of two lines of 40 characters each. It allows for upper and lower case but also some special characters. Unfortunately, 40 characters is never enough and proper punctuation might be the first sacrifice you need to make in order to make things fit. The text is automatically centered to make it look nice on the screen, which is a thing only really noticeable for short sentences.



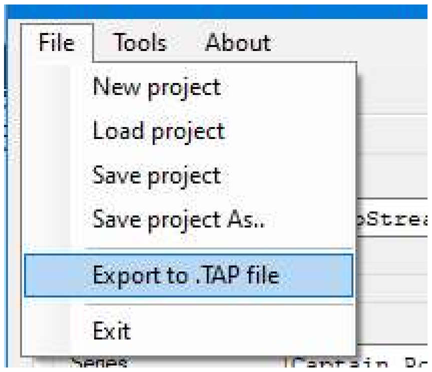
On the right a screenshot of the used character set, this should give you an idea of which characters are used by the program. Characters shown in green are usable for subtitles.

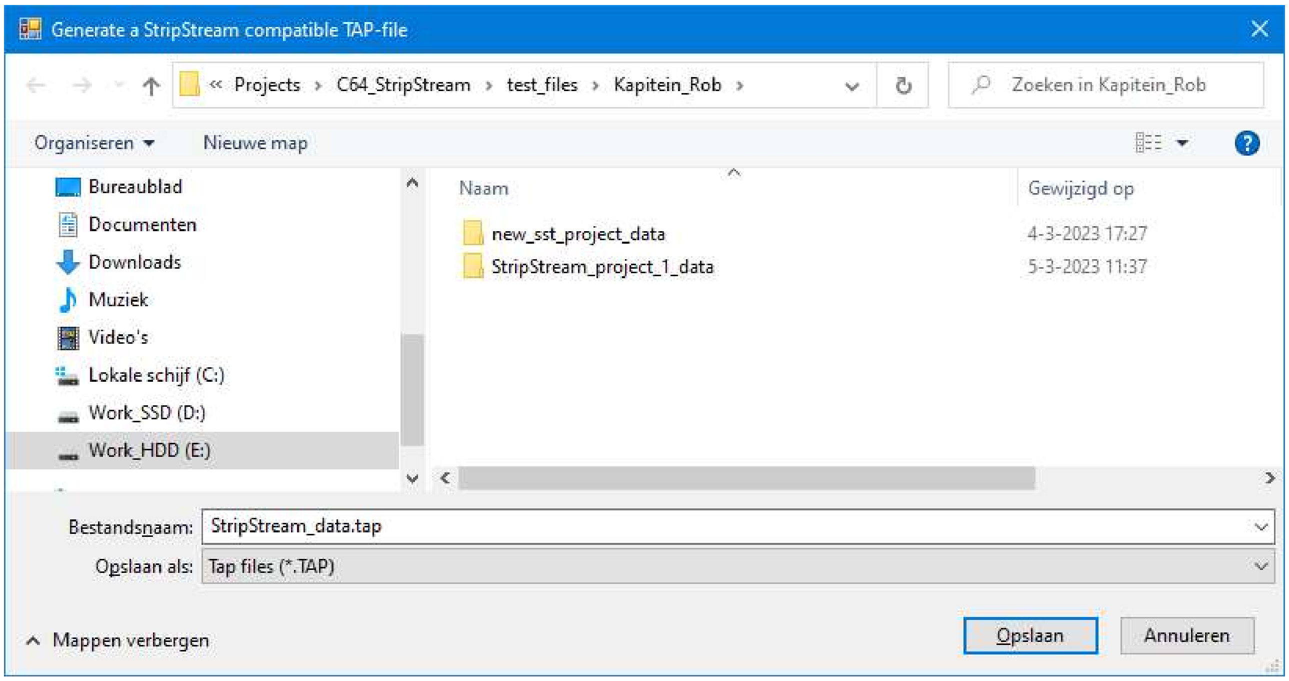
Which language you use is up to you, but always be consistent, if you choose English to be the primary subtitle and Dutch secondary, that's perfectly fine but keep doing that for all frames through the entire story. Otherwise it will be pretty confusing for the reader.

4 Exporting .TAP file

If you want to make you own StripStream release then the StripStream editor program is a one-stop-shop. With one push of the button it creates a file that holds (if configured properly) the player routine at the start followed by the actual StripStream data.

In other words, it creates a .TAP file that you can drag into VICE (the C64 emulator) and it loads automatically and starts playing the stream. This .TAP file might be considered a master which you can use to create a real tape. Go to file and click on the option "Export to .TAP file".





Naam	Gewijzigd op
new_sst_project_data	4-3-2023 17:27
StripStream_project_1_data	5-3-2023 11:37

Bestandsnaam: StripStream_data.tap

Opslaan als: Tap files (*.TAP)

5 Making a tape from a .TAP file

If you have created your own StripStream .TAP file, you might want to make a real tape of that. There are currently various devices that allow you to do this but the most wide spread might be the 1541 ultimate II. However making tapes from .TAP is not a big secret and many information can be found online. I cannot go into lots of detail here, simply because that would require me to name and explain every possible piece of hardware. Instead I limit myself to mentioning the most important pitfalls in tape backup.

A thing to keep in mind, is that ordinary stereo double deck audiocassette players, are not to be preferred to creates retro computing tapes with. As long as you use the same device to play it back with you might be OK. But if you are making a tape in your high-end stereo system, do not be surprised if you get into trouble when playing it back on a real datasette. The main issues are that the datasette is a mono device and expects recordings that are highly saturated. A high-end stereo system tries to prevent saturation and always creates stereo tracks, resulting in a less denser signal. Then there is the problem of the signals being inverted (although that is easily fixed depending on which method you use, for example .TAP to WAV has a selection to invert the waveform). And the problem of head alignment might be an issue, although that's an issue on any kind of recorder, however, hifi stereo systems might be auto-reverse and these kind of decks are not easy to align without the proper tools and experience.

To make a long story short… record your tapes in a 1530 or 1531 datasette.

Although once you have a real tape, you could decide to use that as a master tape which you can duplicate as many times as you like, but if you do so, make sure you do it correctly. These are digital (fully saturated) tape recordings, not HiFi audio tapes and therefore copies should not be made as if they were HiFi audiotapes. Should you want to duplicate your recording by a tape duplicator factory, make sure that you mention that these are not audio tapes and then the fun begins...

To make a long story short, it would be best (although much more time consuming) to make each tape as if it was to be a master tape. Why, because a copy of a master is never as good as the master itself, making analog copies always introduces noise, additional wow, flutter or other speed related issues.

5.1 Test your datasette for basic functionality

How do you know if your datasette is in tip top shape… well this is rather difficult without the proper tools (like a reference tape). But there is one very simple test you can do to test out the most basic form of functionality and that is by doing nothing more then simply using it. Type a simple basic program on your CBM, for example:

```
10 PRINT”TESTING THE DATASETTE”
20 GOTO 10
```

Then save this program by typing the command: SAVE<return>

Then, when the computer asks you to, press record and tape on the datasette and your program will be written to tape. When it's done, rewind the tape and switch of your computer. Then count to 10, turn your computer back on and type the command: LOAD <return>

Then, when the computer asks you to, press play on the datasette and (if all goes well) the program that you saved just before, will be read from tape. If this happens, it shows that the datasette is able to load a small program saved with the kernal loader routines.

Now this doesn't mean that your datasette is in tip-top order and ready to handle everything as you may still experience problems with games that use a fastloader (and most of them do). But when this simple test fails, then there is no way you are able to use it to make a reliable backup to a real tape (or from a real tape) using this device. So that means that you'll need to clean, align or repair it.

5.2 Which tapes should you use?

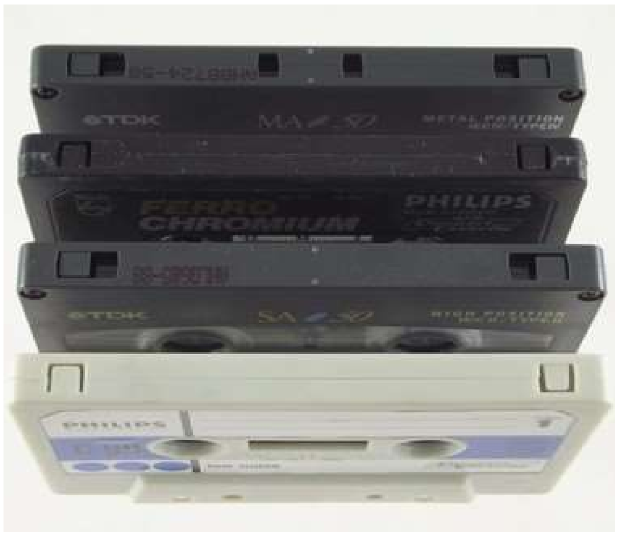
In order to make a good recording you must make sure you are using some decent tapes. If you should choose to use a wrinkly old tape that has been on some attic for 30 years, stored extremely hot during summer and extremely cold during winter under high humidity condition, then be prepared for a very unreliable recording.

Now there is no need to buy the most expensive tape you can find, because that was never the intention of the good people of Commodore, they designed the datasette in a time where tapes where most likely to be very noisy, the high end audio tapes with super low noise and high dynamic range were designed many years later. So to make a long story short, the normal regular (cheap) tape also know as type I is perfectly fine for making you recordings on in a datasette.

In fact, these are exactly the kind of tape you'll should use to get the best results. Because the tapes other then type I require to be written with a stronger magnetic field and those modern kind of tapes, is not where the datasette was designed for. So if you intend to use a "superior" type of tape then in fact you might be making an inferior recording when doing that recording on a datasette. Now this doesn't mean that recordings on these high-end tapes will not work, but you are asking for trouble to make a recording on a tape that was not intended to be used in the 1530/1531 recorder.

Therefore if you plan to make recordings that need to last a long time and/or are distributed to others, then use the proper tapes: Type I (NORMAL) POSITION, NORMAL BIAS 120uS EQ.

In order to provide you with some knowledge about the various types of tape existing and how to recognize them, a short overview below:



Type IV: tapes referred to as "metal", it can be identified by the wider write protect tab hole and extra holes in the center of the tape.

Type III: tape consisting of ferrichrome (FeCr), it can be identified by the markings on the tape only, because it has no extra holes and therefore looks like Type I

Type II: tape consisting of chrome and cobalt, it can be identified by the wider write protect tab hole.

Type I: tape consisting of iron oxide, this tape only has a write protect tab and does not always carry the marking Type I, simply because the first type is never really referenced. Think of the movie Back to the future, Bttf II and Bttf III. When they made the first movie they didn't know they were going to make two more. Really old tapes most certainly won't have this marking, newer tapes might have this marking but most likely won't as it is not a big selling point, when laying on the shelf next to a Type IV. Type I tapes als also referred to as ferro or Fe tapes on the box you'll might find:

Type I (NORMAL) POSITION, NORMAL BIAS 120uS EQ.

Type IV referred to as "Metal" was a completely new formulation. It used pure metal particles instead of metal oxides, hence the name "Metal" as it is based on pure metal and not on metal-oxides. These "metal" tapes were a hard-wearing tape with superior frequency response and greater dynamic range. BUT it also had some disadvantages such as **excess wear on tape heads**, and **expensive** to buy.

Why are there so many kind of tapes?

The main reason for all the various types of tapes was because in the 70's Type I tapes were rather noisy, they had a nasty “hiss” in the background, which was very noticeable when you had some quiet music and the volume turned up quite high. So tape manufacturers started making tapes that didn't have this problem. However, the datasette is not a high-end audio recorder, it is a digital storage device, therefore the small “hiss” noises in the background of the signal are no issue at all for the data recorded onto a tape. So buying a type IV tape sounds nice at first, but does not offer real benefits. Also we may state that Type I tapes produced today (2023) are of a much better quality then the ones made in the 70's.

You might also have noticed that the gaps on the top-edge of the tape are for the cassette-player/recorder to detect the type of tape and therefore set some playback/recording parameters to the correct biasing and amplification/recording values. This tape type detection functionality is not present inside the datasette (the datasette can only detect if the tape is write protected). This because the datasette is designed to work with regular Type-I tapes.

Regarding tape length, never use 90 minutes (or longer) tapes, as these tapes are much thinner and therefore these kind of tapes are more likely to stretch and are very easily damaged. Regular 60 minute tapes are perfectly fine. Although, shorter tapes are most likely preferred if you only intend to store just a single game on a tape. However you can always cut a 60min tape to the desired length by opening the tape enclosure, but that's an exercise for the pure tinkerer and certainly not something for the faint hearted. In essence it comes down to the practical point of preventing unnecessary (re)winding. The longer the tape, the more you’ve got to (re)wind in able to use the other side. A problem easily prevented by choosing the shortest possible tape.

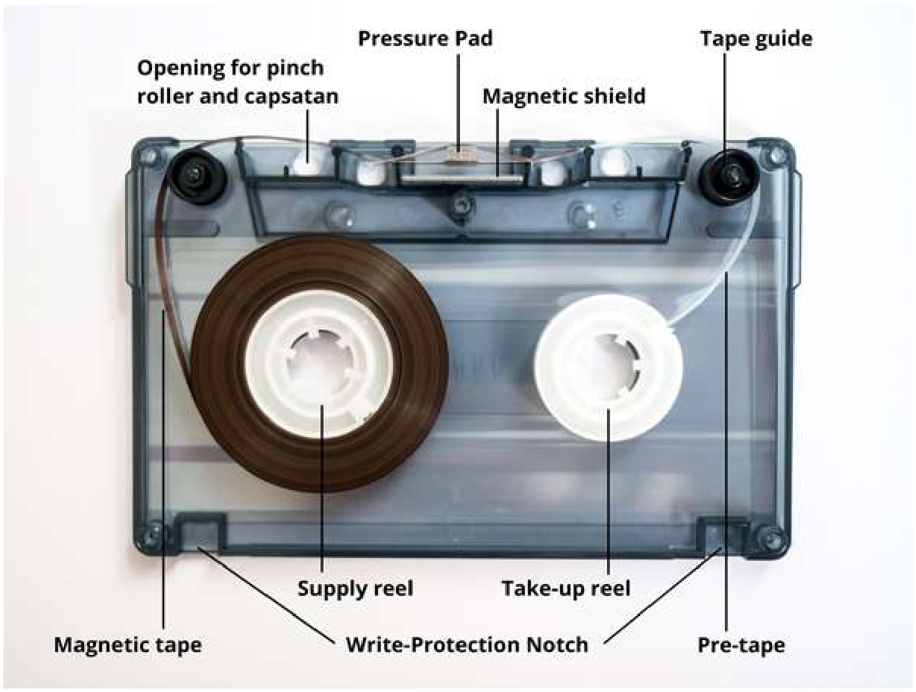
Source:	preferred tape type	page 8 of Compute’s gazette, April 1985, issue 22, vol 3 no. 4
	tape length	page 20 of Compute’s gazette, March 1985

5.3 Condition of source tapes

Although in 2023 you can still buy newly produced Type I tapes, if you only want to make a single copy, then you'll might be tempted to use an old tape you'll have lying around somewhere. Which is perfectly fine. But if you want to a recording on a used tape, you must make sure that the tape you want to create is also able to be read. In other words, don't expect miracles from a wrinkled-up tape.

Make sure that the tape winds and unwinds properly and is that the tape guides and pressure pad in the cassette (the square of foam on a spring that holds the tape against the head) is still in good condition. Because if it isn't, then the tape drifts across or away from the head during recording and playback and it will be very hard to get a write/read a good signal.

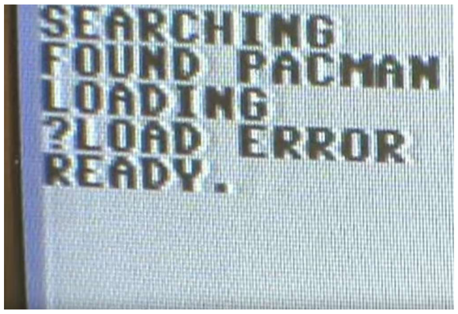
In some cases you can replace the pressure pad by using one from another tape, borrowing it or repairing it using similar materials. And sometimes it's even easier to just swap the reels of the tape into a cassette that has no problems. But those are exercises you'll are more likely to encounter when backing up (preserving an existing) tape, not when creating one. Simply put, if you encounter a bad tape and you want to make a recording, just throw away the bad tape.



Tapes have a write protection notch on their edge. If it is open (removed) a recording cannot be made. If this notch is open, place a piece of adhesive tape to allow for a recording to be made. It may be a good idea, to remove the adhesive tape or (if still present) to break out the write-protect notch of the tape. Because accidentally erasing the content by another recording will be prevented.

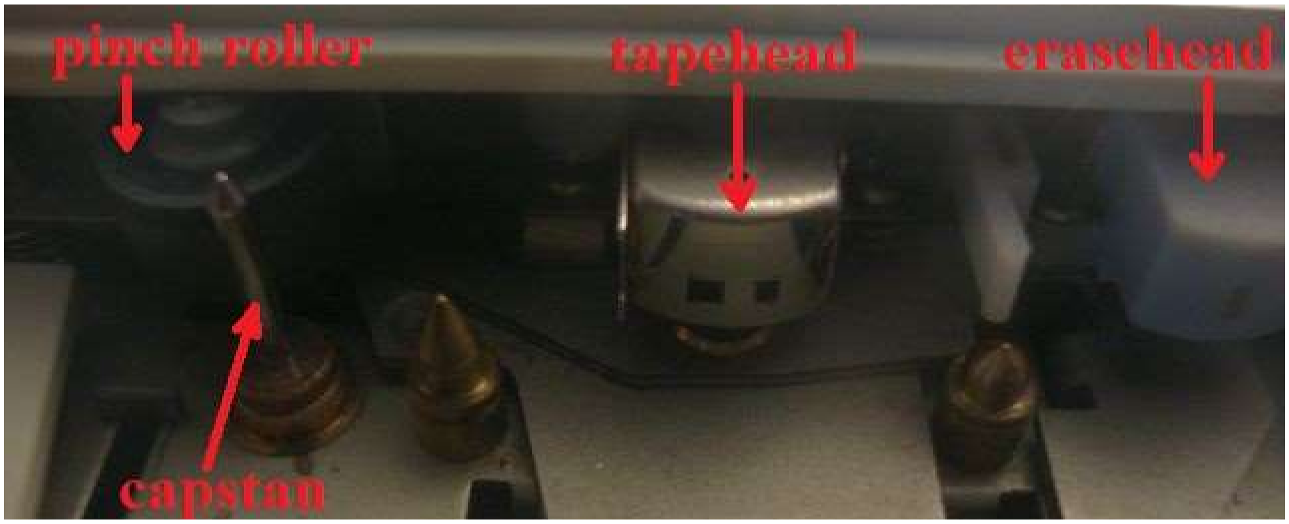
5.4 Datasette maintenance and repair

No matter if you want to recreate/record a tape or want to archive/playback a tape the datasette (or any other type of audio cassette player) must be in perfect order. If not then failure is unavoidable.



```
SEARCHING
FOUND PACMAN
LOADING
?LOAD ERROR
READY.
```

There are many things that can be wrong with your datasette, which is to be expected for equipment of this age. Sometimes it is only a matter of good cleaning of the rubber wheel called "pinch roller" and the tape heads. But sometimes it may be required to replace all parts that are "rubber". A lot of times the belts are not a tight as they where when they left the factory and therefore they are slightly loose and cannot transfer the power from the motor to the tape mechanism, causing problems during playback or (re)winding. One of the most common problems with the datasette (and any other kind of tape drive used for home computing in the 80's) is tapehead alignment, also referred to as azimuth. Make sure that your tapehead is aligned properly. Below an image showing the easy accessible and important parts of your datasette.



So in other words… always make sure that your datasette (or other kind of tape recorder) is in perfect working condition. A backup or restored tape is only as good as the recorder is was made on. If you use a crappy recorder, you'd get crappy recordings!

Finally there is one thing that should be taken into account, use proper tapes, tapes that are 30 years old and have been subjected to moisture and continuously varying temperatures are not really a good base to be used for restoring your tape archive from .TAP file to real tapes. Therefore make sure you are using tapes that are in top condition.

5.4.1 Belt replacement

Worn out, loose, broken or sticky belts will make it impossible for the tape to be played at a correct or constant speed. Worn out (or loose) belts inside the tape mechanism can create problems with the continuity of the speed of the tape. You will most likely notice this during winding and rewinding, the tape sometimes stops somewhere close to the end or the beginning of the tape. You must fix this in order to continue. Belt problems can be detected (sometimes) by carefully listening to the sound of the leader, the long BEEEEP tone at the beginning of each game/program on the tape. Place the speaker switch of the Cassiopei in the playback position and playback your tape. When the leader sounds like PEEEEEEEEEEEEEEEEEEEEEEEEEEP, your probably OK.

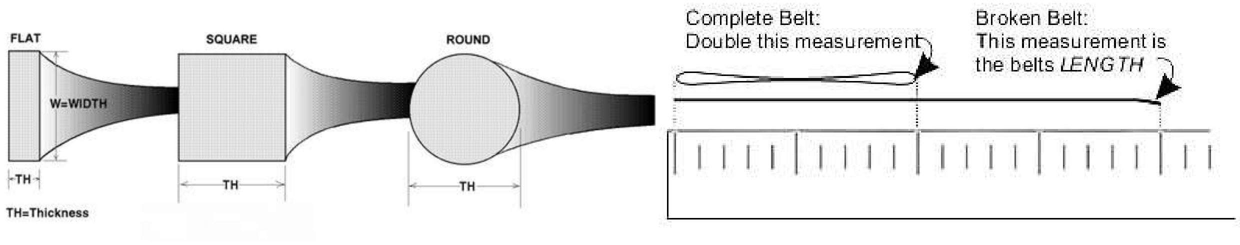
But when it sounds like BWWWWEEEEEEEWIIIIIWUUUUWEEEEWIIIWEEEWUUP (or something similar) then your are in big trouble. Open up your datasette and inspect the belts. The belts must be perfectly straight, if there is a bump in the belt, then this bump will affect the tape speed. These bumps are caused by the belt being in the same position for many years when exposed to the elements of attics and garages. The high temperatures in the summer and the high humidity of winter simply isn't ideal for any material.

Attention: when performing this quick test, make sure that you have a tape from a reliable source, do not use a tape you made yourself, use a commercial game tape as a reference. As these kind of tapes are made with high-end recording equipment with very low wow and flutter. So there a “PEEEEEEEEEEEEP” is recorded with a perfectly constant speed, which is just the way we need it to be in order for it to act as a reference.

Tip: sometimes (many times they can't), loose belts can be revived (shrunk). Take a pan, add 200ml of water, boil the water, switch of the heater, place the belt in the hot water. Leave it there for a minute, then carefully take it out (best is to dump the contents of the pan into the sink).

Tip: the perfect way to solve your belt problems would by by replacing the old belt for a new one. However, when ordering a new belt make sure that it is the right one as lot's of things can go wrong. Do not be tempted to buy a bag of general purpose belts from ebay and hope to find a belt that almost fits, although this looks OK at first, you may encounter problems with the tape speed when the belt is too tight, you may encounter problems when the belt is too small then it may not fit properly in the pulley of the motor, it might even get stuck or wobble. Also if the quality of the belt is bad, the thickness may vary. This will immediately result is speed variations every time the thick parts passes the motor, these problems are easily heard during the “PEEEEEEEEEEEP” test. And cause the effect called “wow”. Describing the exact type of belt for the datasette, is beyond the scope of this manual as there are many kinds of datasettes made by various manufacturers. Please refer to the service manual of your datasette for the exact type. Or carefully measure the old belt (thickness and diameter, see how to do this on the next page). Please be aware that an old belt MIGHT (it doesn't has to) be a littlebit stretched, so please, on ordering a new belt, allow for a correction of a few millimeters smaller belt. A belt is most certainly stretched when it starts to slip.

Below, the method of measuring the belt size, type and thickness. Always use the same type!

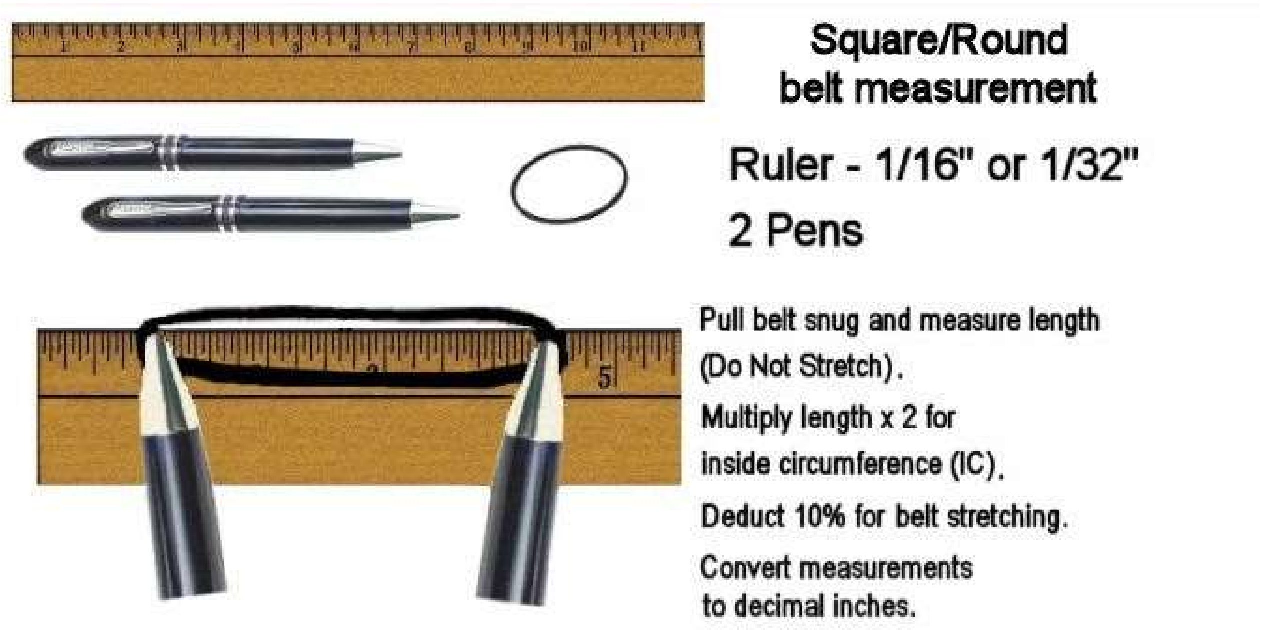

FLAT
W=WIDTH
TH
TH=Thickness
SQUARE
TH
ROUND
TH
Complete Belt: Double this measurement
Broken Belt: This measurement is the belts *LENGTH*

When I (the writer of this manual) replaced the belt in my 1530 datasette, I measured that the thickness of the new belt was 1mm x 1mm (in other words, a 1mm square belt).

It had a folded size of 122mm (or a length of 244mm)

I also measured the diameter of the belt, when I laid it out in a nice circle, that measured 78mm. Now these measurement may be a little bit inaccurate, considering that it is a material that stretches easily, but choose the belt that is closest to you're measurements and you'll be fine. Keep in mind that the old belt might be stretched a littlebit and therefore measure a slightly larger size (which could be up to 10%), this may not seem like much… but it's more then enough to cause problems.

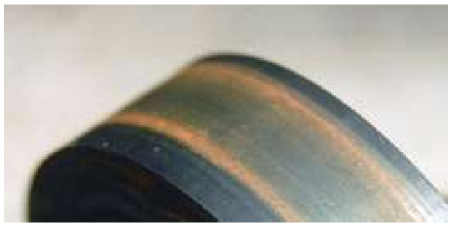
Another way could be by using a simple ruler and two ballpoint pens, a method shown below:


Square/Round belt measurement
Ruler - 1/16" or 1/32"
2 Pens
Pull belt snug and measure length (Do Not Stretch).
Multiply length x 2 for inside circumference (IC).
Deduct 10% for belt stretching.
Convert measurements to decimal inches.

5.4.2 Pinch roller problems

The pinch roller pushes the tape against the capstan (the metal pin that is actually moving the tape) the capstan is directly connected to the flywheel which is driven by belts.

The pinch roller could be dirty, in very severe cases it could be even as dirty in as shown the picture on the right. You can clean it using a cue-tip or cotton swaps and some alcohol (for instance medical wound cleaning alcohol).



Put the datasette, without a tape, in the play mode and gently clean the now rotating pinch roller. If you notice that the capstan is dirty too, clean that as well. Keep cleaning until all the dirt has been removed.

After cleaning wait at least 15 minutes for it to properly dry before you start playing tapes. Otherwise the tape might stick to the capstan and will be damaged.

If the pinch roller is very smooth and shiny, like a mirror like surface shiny, then it may be helpful to roughen the pinch roller a little bit, some fine sandpaper can do the trick. While doing this, make sure you'll sand the surface evenly, because you don't want any flat spots. Therefore this should only be tried when there are no other options left.

When the pinch roller has hardened (which is not unlikely to happen over a period of 30 years) then it won't push against the tape with the correct force. Causing the tape to slip, causing the tape speed to be irregular. When the pinch roller can be pushed in with your fingernail and bounces back you're OK, if it doesn't, you'll need to replace it.

Another problem might be that there is a dent in the pinch roller. If for some reason the datasette has been stored for a long time with the play button pressed, then the pinch roller has been pressed onto the capstan for a long time, causing an impression in the rubber pinch roller that won't go away. The only option is to replace the pinch roller as it is permanently damaged by the constant pressure of the capstan. Fortunately, pinch roller of the type as used in the 1530/1531 are still available and do not need to be expensive.

5.4.3 Tapehead cleaning

Sometimes things are just dirty and need a good scrubbing. Your tapehead could be really needing this. You can do this using cleaning alcohol and some cue-tip or cotton swaps. Gently dip the swaps in the alcohol and move them across the tapehead in a gentle but firm sweeping motion. Repeat this as long as there is dirt coming of the tapehead. Don't forget to use new cotton swaps when they are becoming dirty, you don't want to spread the dirt over the head, you want to remove it.

Don't forget to clean the erasehead too, although you don't need it for reading back tapes it is very important for writing. When cleaning of both heads is done, make sure to let it dry for at least 15 minutes before using it, you don't want your tapes to get wet OR to stick to your wet tapehead while recording/playing.

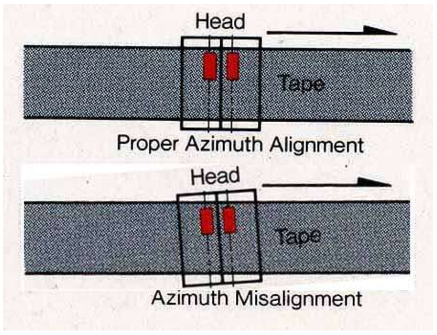
5.4.4 Tapehead (azimuth) alignment

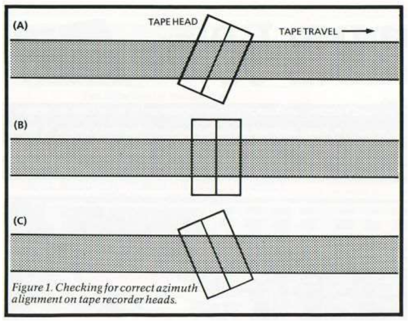
Calibration of the tapeheads regarding the angle of the head relative to the tape (azimuth) is a big problem. An incorrect angle will make a perfect tape impossible to load. But also a tape saved at the wrong angle will be impossible to load.

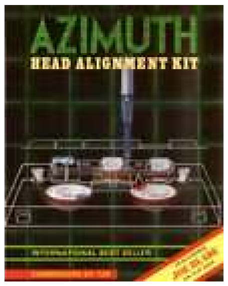
Tapehead alignment is required to make sure that your tapehead is capable of reading the tape signals correctly. The first thing that suffers from tapehead alignment is the high frequency response. In other words high frequencies on the tape cannot be properly detected, if it was an audio tape then the sound would sound less crisp on a bad alignment. Some would say that the sound would sound muffled. But considering we can't hear what the datasette outputs were slightly in trouble. We can't adjust the tape “by ear”.

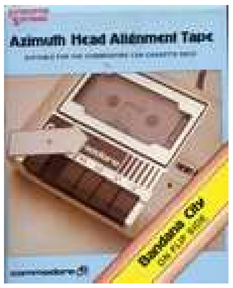
This problem mainly occurs when using tapes that have their data stored using a fastloader. Fastloaders use smaller pulses (and some other tricks) and therefore can send their data more quickly. However these smaller pulses simply mean higher audio frequencies and that is when the importance of a perfect azimuth becomes clear. Because if your tapehead isn't properly aligned then it can't follow/detect the higher frequencies of the tape correctly, causing data errors leading to the infamous “LOAD ERROR” message. Of simply a game/program being unable to load...

But first, what is azimuth? As the image below shows, a perfect azimuth is when the angle of the tapehead is identical to the angle of the tape. This misalignment can be in 2 directions.



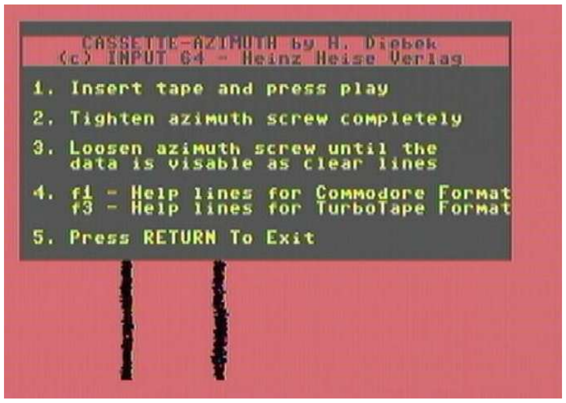

(A)
TAPE HEAD
TAPE TRAVEL
(B)
(C)
Figure 1. Checking for correct azimuth alignment on tape recorder heads.





The only way to solve this problem is to make sure that your tapehead is aligned properly, this can be done by adjusting the position of the screw slightly. There were special alignment kids back in the day that would have a tape with a calibrated signal to which you aligned your tapehead to. There was even a special datasette clone, called “LOAD IT”, this model had an LED bar display, indicating the audio level, allowing you to visualize what you otherwise can't hear. But chances are that you don't have any of the items shown here.

Fortunately, you are not alone. And in practice there isn't a special calibration tape required. All you need is a tape with a signal on it written in a time where the calibration was still OK. Or written on another device that has a good/perfect alignment. The best you could do is using the tape you intend to load, that way you would get optimal alignment for that tape, which could be better (for that situation) then the perfect azimuth. Regarding a tape to use to adjust your tapehead to the perfect azimuth, you can use most commercial games tapes. Because these tapes were written on machines with a good/perfect head alignment.



```
CASSETTE-AZIMUTH by H. Diebek
(c) INPUT 64 - Heinz Heise Verlag

1. Insert tape and press play

2. Tighten azimuth screw completely

3. Loosen azimuth screw until the
   data is visable as clear lines

4. f1 - Help lines for Commodore Format
   f3 - Help lines for TurboTape Format

5. Press RETURN To Exit
```

On the right, a screenshot is shown of a program for the C64, it is called "recorder justage" (it can be found on the internet on many places, for instance on CommodoreScenDataBase (CSDB.dk)). This tool is perhaps the most used tool for tape adjustment, simply because it was released for free in a magazine, the same program can be found under various names as it has been slightly modified by some or translated. But the essence of all is the same, make sure the shown lines are as thin/clear as possible. The location of the lines may vary, depending on the type speedloaders the tape uses. Although it is not required to use a tape that has a speed loader, in fact the actual data on the tape doesn't really matter, as long as it is there long enough for you to calibrate the tapeheads. If the tape is too short, you'll be constantly rewinding the tape, which could be a bit unpractical.

The story behind the program "recorder justage" (according to people on CSDB) goes as follows:

Recorder justage by Harald Diebek, from Input64 06/1985.

Harald owned a computer store, one of the top seller was the C64. He had a lot of returns and complaints from customers who had problems loading tapes (which they obviously copied from a neighbour or friend). Harald got tired of all the complaints and grabbed a "C64 intern" and coded this little tool. Some time later he eventually sold it to the "heise verlag" for publishing on "input 64" for 3000DM. For that money he bought himself a PC-XT. Harald never had his own c64 (at home that is), and he never programmed any c64 stuff again.

How to use the program? All you need to do is play the tape back and observe the signals as a function of the time (you could call it a spectrum of the audio signal or you could call it a waterfall display). But what's important is that the screen showing the signals must show lines that are as sharp as possible which is only the case when head alignment is optimal. Insert a tape into the datasette, fully rewind it and press play, then insert the screwdriver. Therefore watch the screen while gently tightening the screw by turning the screwdriver clock wise until the signals get worse. Mark this position. Turn your screwdriver counter clockwise until the signals get better and keep turning until they get worse. Mark this position, now turn your screwdriver to the position exactly in the middle of the two marked points. This should be the most optimal position. Remove the screwdriver and stop the tape, you are ready to load.

A similar program, but without a fancy screen is written by “Onslaught”, it is called “headalign.prg” but it is basically the same thing. It can be downloaded from the Commodore Scene Data Base (CSDB.dk)) or simply typed in (see code below) for the ultimate retro experience.

The code was kept as small as possible, because the requirement was that user could type it in. Here is the code below, so you could type it in for your C64, which is quite practical if you have no device to load it from.

```
0 REM ***  VISUAL DATASETTE HEAD ALIGN TOOL BY ENTHUSI/ONS ***
1 D=0: FOR B=0 TO 196: READ A: POKE (4*4096+B),A: D=D+A: NEXT:
2 PRINT D:IF D=26242 THEN SYS 4*4096
10 DATA   32,  68, 229, 169,  59, 141,  17, 208, 169,  24, 141
11 DATA   24, 208, 138,  41,   7,  10,  10, 133, 252, 138,  74
12 DATA   74,  74, 133, 253,  74, 102, 252,  74, 102, 252, 101
13 DATA  253,   9,  32, 157,   0, 192, 165, 252, 157,   0, 194
14 DATA  189,   0, 194, 157,   0, 194, 144,   3, 254,   0, 192
15 DATA  232, 224, 200, 208, 209, 169, 128, 133, 252, 162,   0
16 DATA  138,   41, 248, 157,  0, 196, 165, 252, 157,   0, 198
17 DATA   74, 144,   2, 169, 128, 133, 252, 232, 208, 235,  88
18 DATA  160, 254, 204,  18, 208, 208, 251, 173,  17, 208,  48
19 DATA  246, 173
20 DATA  173,  13, 220,  41,  16, 240, 249, 173,  13, 220, 162
21 DATA    0, 169,  16, 232,  44,  13, 220, 240, 250, 120, 165
22 DATA    5, 201,   3, 176,  22, 164,   6, 185,   0, 194, 133
23 DATA   20, 185,   0, 192, 133,  21, 188,   0, 196, 177,  20
24 DATA   29,   0, 198, 145,  20, 198,   5, 208, 203, 169,   6
25 DATA  133,   5, 198,   7, 208, 182, 169,   5, 133,   7, 230
26 DATA    6, 165,   6, 201, 200, 208, 170, 169,   0, 133,   6
27 DATA  133, 195, 162,  32, 134, 196, 168, 145, 195, 200, 208
28 DATA  251, 230, 196, 202, 208, 245, 240, 147
```

Insert a tape into the datasette, fully rewind it and press play, then insert the screwdriver. Therefore watch the screen while gently tightening the screw (turning your screwdriver clock wise) until the signals get worse. Mark this position. Turn your screwdriver counter clockwise until the signals get better and keep turning until they get worse. Mark this position, now turn your screwdriver to the position exactly in the middle of the two marked points. This should be the most optimal position. Remove the screwdriver and stop the tape, you are ready to load.

5.4.5 Tape speed

Making a recording in a device that runs 5% too fast and playing back that same recording in another device that runs 5% too slow, will result in a total signal error of 10%. Which is not good! Now although the StripStream protocol can handle a tape speed error of 10%, you should never attempt to make a recording in a recorder that is not functioning properly. Although you not always have control over the device used for tape playback, you do have control over the device that makes the recording. So by making sure that the recording device runs on the perfect speed you can limit the possible speed related playback error to only the error of the playback device.

In order to adjust the speed of your datasette, you'll need a reference tape and a program to measure the signal. I have written a program to do such a thing and made a video about it, for more info about tape speed, please go to:

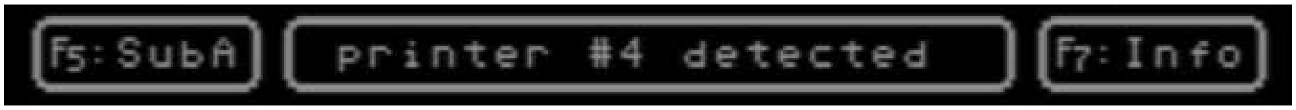
https://janderogee.com/projects/1530_tapespeed/1530_tapespeed.htm

6 Printer support

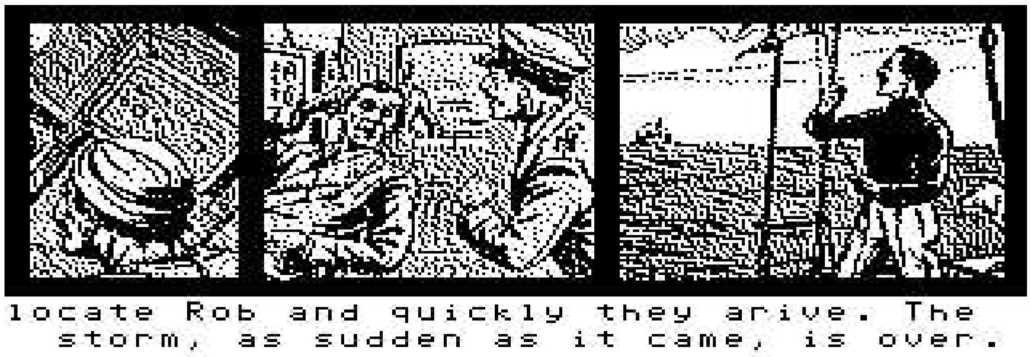
For those who prefer to read their books from paper, just buy the book. But just because it's slightly silly to print a digital version of a comic book onto paper, printer support has been build in the StripStream player. It's functionality is hidden, normal (printerless) users might never even notice its existence. Printing functionality effectively is an easter egg, not very well hidden though, just switch on your printer and it works. However, it's nothing fancy. Basically it's a screendump of the complete image section and the corresponding visible textfield.

Only MPS803 compatible printers are supported, fortunately many printers are MPS803 compatible. However, it is required that the printer is connected via the IEC serial bus and that the device is set to act as device #4. Make sure that the printer is switched ON before the StripStream program is started.

When the StripStream player is started the first thing it will do is check if the printer is present, if it can detect the printer, than printer support is enabled and every new image loaded (including subtitle), will be send to the printer. This will be indicated by the screen as shown below.



When no printer has been detected at startup, there will be no printer support at all. If you don't want you printer to print anything, simply switch the device off. If you do want to print, the results you may expect will look like the example below (VICE printer emulation), keep in mind that the result on a real printer will not look as crisp as shown below. Dot-matrix printers aren't known for their image quality.


locate Rob and quickly they arive. The
storm, as sudden as it came, is over.

The C64 is a powerful beast, well in the early 80's at least, but it isn't always easy to program and printers are just the same. Keep in mind that this StripStream printer functionality is merely intended as a joke, considering that StripStream is a digital comic book reader, why would you ever want to print. Seriously, why would anyone in 2023 want to printout a comic book on a dot-matrix printer. It's expensive (ink ribbon), IT'S LOUD, printing quality sucks and it takes hours to print it all. So you can imagine that with this in mind not too much effort was put in the printer routines. The image that is printed is only a screendump of the Hires screen and text field. The printout isn't scaled or rotated, so it doesn't fit the paper in the most optimum manor.

Printing will take place once the image has been loaded completely and when the user is required to push a button to continue loading the next image. For anyone who do wants to print the entire contents of the tape to paper but doesn’t want to stay near the computer for several hours of printing time, just press shift-lock, this is identical to the user pressing a button. This because shift-lock is held down mechanically so the button is continuously pushed, allowing for automatic playback of the story.

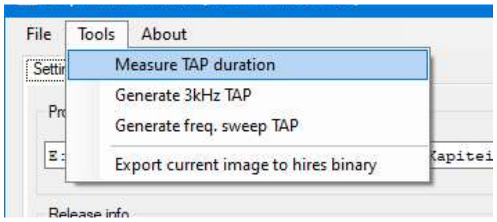
Some people might wonder why the entire image section of the screen is printed, while it would seems so much more logical to only print the last image. The main reason is simplicity. Writing a printer driver on a C64 isn’t easy and requires lot’s of code. The mian issue is that the alignment of the image data in C64 memory isn’t identical to the way it should be printed to the screen. With the exception of the first pixel of the image, nothing matches. So all the data needs to be reformatted when send to the printer. Personally I would have preferred to have printed all the images rotated by 90 degrees, so that the entire story can be printed as a stream to a piece of endless (80’s style) printerpaper, allowing the book to be a true strip of tens of meters long. But, not all images are the same size, so where would I put the subtitles, how would it all align nicely? It would never have the charm of the screen and always be a compromise. Therefore I decided to keep things simple and concentrate on to the StripStream project itself, focusing on the reliability, speed and presentation of the StripStream data reliably onto the TV/monitor). So, therefore the printing to paper routines are functional but very rudimentary. And let’s be honest… who’s ever going to use it?

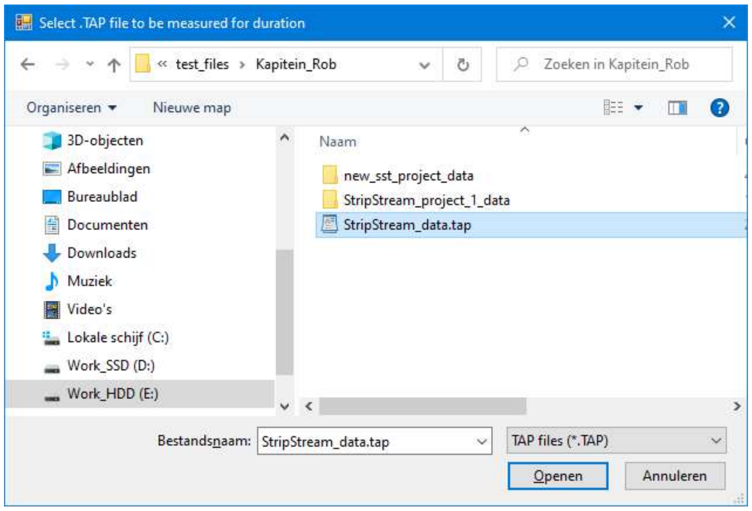
7 Tools

The StripStream editor program has some handy tools. Although these tools are not to be expected to be used by the general public.

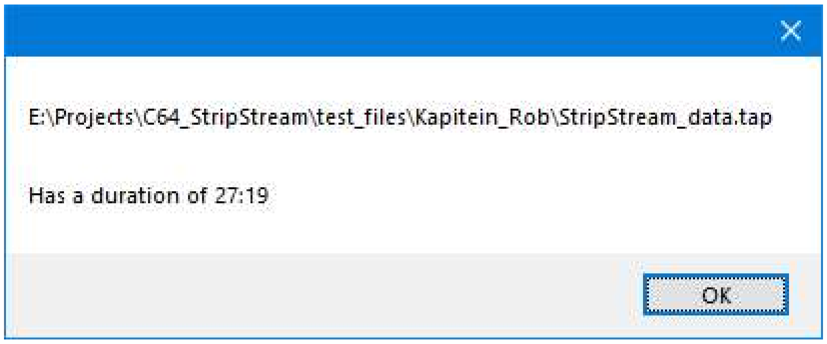
7.1 Measure .TAP file duration

The “Measure TAP duration” tool, allows the user to select a .TAP file and calculate its duration. Because sometimes you are just curious how large the end result has become or if the splitter has split the file at the desired length.





After having selected the file of interest, the systems starts to measure the file, this might take a few seconds depending on the speed of your computer and the size of the file. The result is presented via a popup screen, see example below.

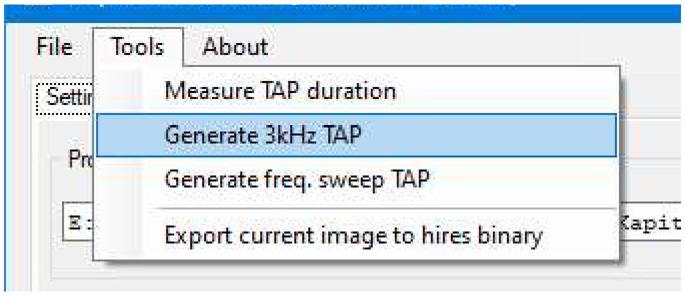


E:\Projects\C64_StripStream\test_files\Kapitein_Rob\StripStream_data.tap

Has a duration of 27:19

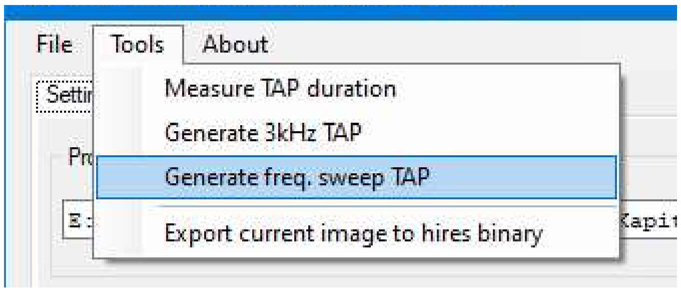
7.2 Generate 3kHz TAP

This is a very simple signal, it has no use other then for analyzing the performance of the datasette. It was used during the development of the StripStream protocol, however it could be of use to somebody else, so it was left in the program. It does exactly what it says in the description, it creates a .TAP file holding nothing but a 3kHz tone. The duration of the generated .TAP file will be approximately 35 minutes.



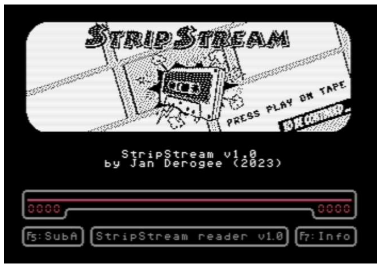
7.3 Generate freq. sweep TAP

This is a very simple signal, it has no use other then for analyzing the performance of the datasette. It was used during the development of the StripStream protocol, however it could be of use to somebody else, so it was left in the program. It does exactly what it says in the description, it creates a .TAP file holding nothing but a frequency sweep tone that ranges from 5132Hz to 838Hz and will repeat this sweep for about 100 cycles which means a total duration of about 5 minutes.

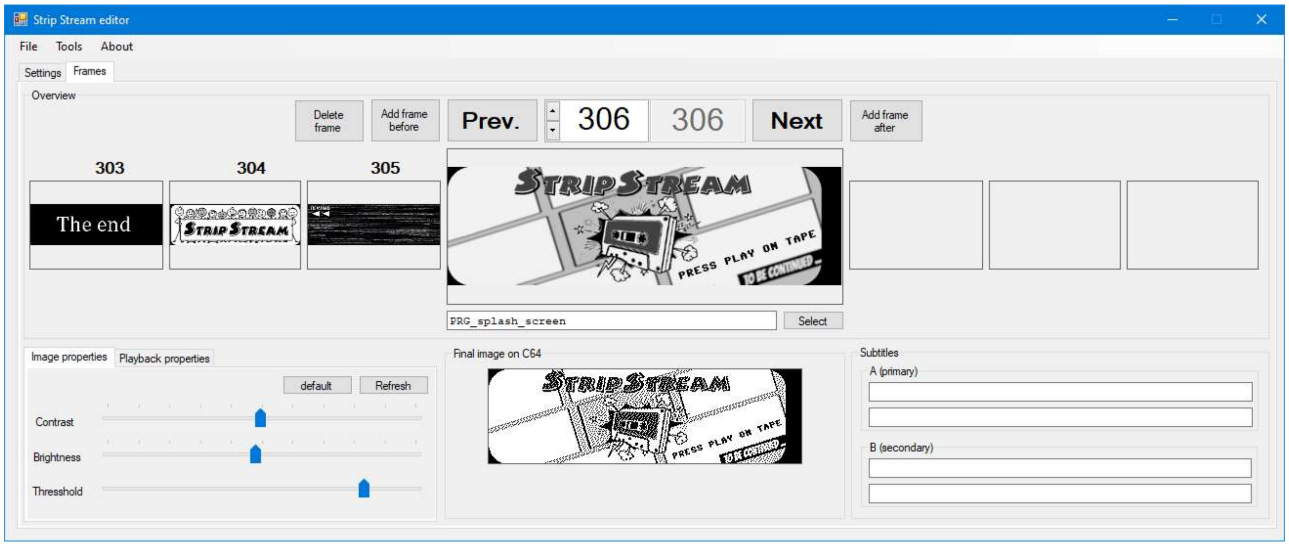


7.4 Export current image to hires binary

This is a tool that's really only useful for creating the splash screen of the StripStream player program. On the right is shown a screenshot of the splash screen when the StripStream player starts. This this picture is embedded into the StripStream player program and the dataset for that hires image was created using this tool.

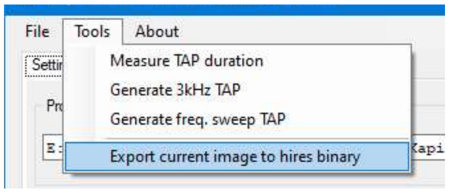


It works pretty basic, just import the splash screen image (which must be 320x96 pixels (or a multiple)) into the program as if it was an ordinary frame, adjust the contrast, brightness and threshold settings to the appropriate levels.

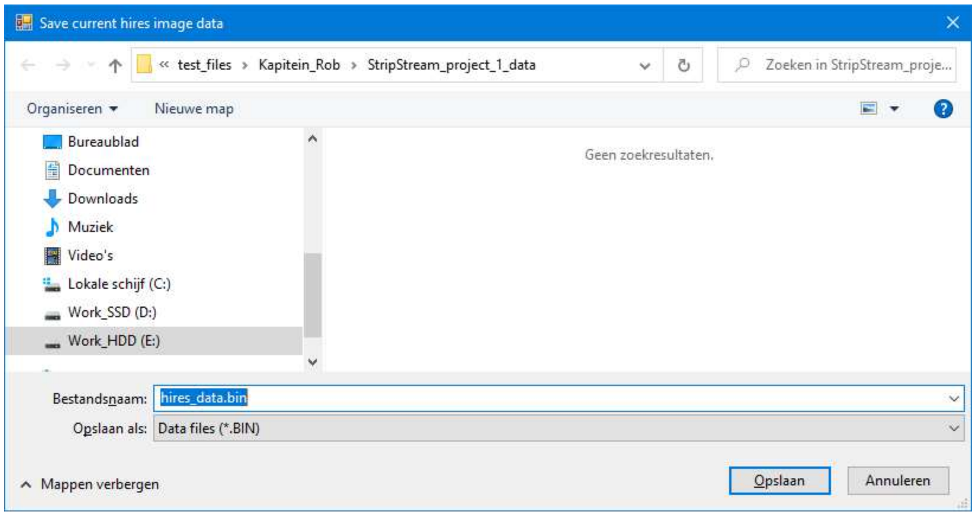


The image shown in the editor program, looks slightly different then in on the C64. This because the image in the editor, has just like the real image, a 16 pixel black bar on the right side of the image. This is very important, because it is required for proper alignment on the C64 screen. Although because of scrolling the practical (visible) screen width is only 320 – 2 columns of 8 pixels = 302 pixels wide, but since the image still needs to align width a memory layout of 320x96 pixels we have a part of the image that will not be shown. Which is created with the 16 pixel wide black bar on the right side of the image. Keep in mind that that black bar will not be visible on your C64.

When the image looks like it should look on the C64, go to the tools menu and select the option "Export current image to hires binary".



A file requester will appear and it will suggest to save the data under the filename “hires_data.bin”.



Bestandsnaam: hires_data.bin
Opslaan als: Data files (*.BIN)

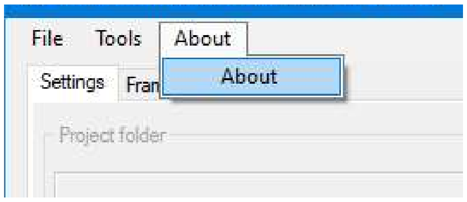
The generated file will be holding the hires screen data that can be directly included into the StripStream player source code. Which was programmed in assembly using CBM Program Studio.

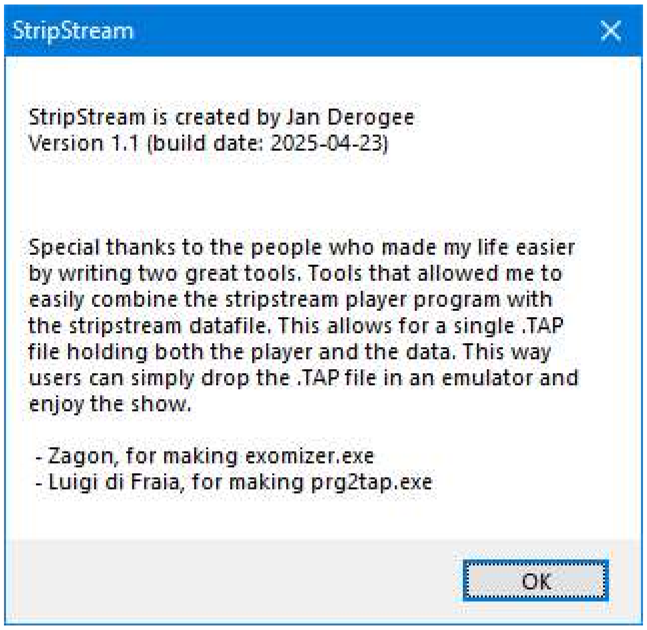
```
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
;                                              BITMAP (screen-A)
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
;This is screen-A ($4000-$4EFF), place a bitmap here.
;It will be shown when the program starts and replaced as soon as the data from the tape starts coming in

*=$4000

;Place a binary file of an image exactly 320x96 pixels (this file holds the hires data of the splash screen)
incbin "data_hires.bin",0 ;value is the number of bytes to be skipped from the beginning
```

To prevent confusion, this export tool is merely for development purposes, normal users will not have any use for this function.

8 About

The about screen is just a simple place to show the credits.





StripStream is created by Jan Derogee
Version 1.1 (build date: 2025-04-23)

Special thanks to the people who made my life easier by writing two great tools. Tools that allowed me to easily combine the stripstream player program with the stripstream datafile. This allows for a single .TAP file holding both the player and the data. This way users can simply drop the .TAP file in an emulator and enjoy the show.

- Zagon, for making exomizer.exe
- Luigi di Fraia, for making prg2tap.exe

Mentioned in the credits are Zagon and Luigi di Fraia, although they did nothing related to the StripStream concept itself, they've do made the tools that allowed me to include the StripStream player program into the final StripStream data .TAP file. Which makes the whole concept of the creation and usage of StripStream tapes so much more practical. Allowing for a .TAP file that can simply be dropped into an emulator and works without pressing any further buttons. Drag, drop and enjoy the show. Or on a real machine, insert the tape, rewind to the beginning, type LOAD, press play on tape and enjoy the show.

So thank you Zagon for making the tool "Exomizer". Which is a tool I use to compress the player program (program.prg) into a much smaller program. Exomizer is perhaps one of the most used compression tools in the Commodore 64 retro computing scene, simple because it works so remarkably well. It's small, it's fast, it's great! And it helps reducing the loading time of the player program significantly making things so much more practical.

Ad thank you Luigi di Fraia for making the tool "prg2tap". This tool was needed in order to add the loader program to the front of the .TAP file. Because the loader needs to be loaded with the standard Commodore kernal loader, which is a completely different loading protocol than used for the transfer of the StripStream data. The tool "prg2tap" converts a .PRG file into a .TAP file. And by combining this .TAP file with the data .TAP file, it is possible to create one big .TAP file that holds both the player and the data. But where the "prg2tap" tool really shines is the ability to apply a fastloader. Meaning that the loading times of the StripStream player program could be decreased significantly.

So thanks to the tools "Exomizer" and "prg2tap", StripStream tapes can include the player program at the start of a StripStream tape while only consuming about 40 seconds of tape space.

9 How it all works...

The next few chapters go in fine detail in how the StripStream protocol works, what packages are used, etc. All that information is not for the common user, it is information for programmers, information required in order to understand the workings of the concept. This way others might learn from this project and for me (the author) it's the only way of archiving in a proper way, what I made and how I made it. Allowing this project to be fully open source so that others can build upon it.

Scrolling HIRES images, takes time… and a lots of it. Because HIRES images simply require a lot of data, and although the C64 does have hardware scroll registers, these only allow the scrolling of the screen on a sub-character level. Meaning it cannot shift the screen more then 7 pixels max. After that the entire screen needs to be scrolled 8 pixels to the side, reset the hardware scroll to position 0 and then the process repeats. Scrolling the entire screen one character to the left would require 320x200 pixels / 8 pixels per byte = 8000 bytes to be copied! This takes a lot of CPU time, because the C64 doesn't have any memory processing functionality other then moving the data one byte at a time by the CPU itself. And all this time, the tape keeps on playing. So the only solution to that would be that the data on the tape should allow a period of no data, so that the copy action can be performed without loosing data from the tape.

Bad-lines, this simply mean that every 8th line (if the screen is enabled) the VIC (C64 video chip) blocks the CPU for a period of 40-43 cycles. During that time the CPU simply cannot operate, so it cannot detect signals from the tape and cannot perform any interrupts. In other words, when the screen is enabled (which it needs to be otherwise there is simply nothing to see). We will be experiencing this problem every 512us (1/8 of the 15625Hz video line frequency). Now this does not have to be a problem, as long as we make our tape signals long enough to fit one or multiple of these "CPU blackouts".

And last-but-not-least there is the split screen, because we do not want the entire screen to be hires, we also want to be able to put some text onto the screen. Which is also an area on the screen that is not subjected to the hardware scroll of the HIRES screen. Therefore a split-screen or raster interrupt will divide the screen into two sections, where the top section is HIRES and the bottom section is TEXT. The splitting of the screen takes place at a line that is in the middle of two bad-lines, preventing the problem of an interrupt happening before or directly after a bad-line (which would otherwise be causing the maximum loss of available CPU time).

Now then there is the tape and mechanism. The tape is most likely of a bad quality, the datasette may be in a bad shape, significant wow, flutter and bad alignment of the tape head is to be expected. Dropouts on the tape or even a bad copy of the tape is to be expected.

The datasette is a fun device, a dedicated audio-cassette player intended for the recording and playback of computer data and computer data only. Although the device accepts compact cassette audio tapes, it cannot playback analog audio. The system uses saturation recording, there is signal on the tape or either there isn't. But you can't just send data directly to an ordinary tape, because what would happen if you'd send a long series of 1's or 0's. Then there would be a problem, because the tape-head can only detect fluctuations in the magnetic field on the tape. And the amplifiers are AC coupled. In other words, we need to send a signal that continuously alternates between ON and OFF. We need some kind of guaranteed modulation.

One way to achieve this is by sending tones or pulses to the tape. A frequency of 20kHz would allow the transmission of 20kbits/second. But although that sounds reasonable, it is complete nonsense (or bullshit to be more precise) for this kind of device. First of all, it isn't high end audio, second of all 20kHz is not really possible for regular audio tape moving at the standard speed. And the datasette has a fixed speed, the same speed of an ordinary audio tape deck, why… because that was cheaper. And speed wasn't an issue when this system was invented, a PET with only few kByte of data will not be loading large/long programs anyway. It only became a problem when games of 50 kBytes needed to be loaded into the memory of a C64, but by then it was already too late to change. And seriously, if you want speed, buy a diskdrive. Commodore developed the tape protocol with reliability in mind, and with that concept in mind they passed with flying colors.

The datasette is designed to be used in combination with type-1 tapes (fero tapes) fortunately modern 2022 tape production of these kind of tapes is still existing AND of much better quality than 50 years ago. Then there is the problem of a 10% variance in the tape speed. So technically, if we want to have a reliable playback/recording we are limited to a pulse frequency of max. 5kHz (which is a value not uncommon in the world of data storage on regular audio tape. This would allow for 5kHz or 5kbits/sec of data storage you would think, but think again, how would you detect the difference between a stopped tape and a series of logical 0's? So there still needs to be a signal defined for a logical 0 and a signal defined for a logical 1. It is not uncommon to choose 5kHz for a 1 and 2.5kHz for a 0, which would allow for 2.5kbits/sec of data transfer. But… that's not it, we also need some form of synchronization, because how would we know when a byte has started or stopped, so additional signals do need to be added.

Then there is the fact that the C64 cannot detect levels of the tape, it can only detect the falling edge of the tape signals. In other words, we cannot detect the width of the high or low period, we can only detect the time between one falling edge of a tape signal and another. This makes things slightly less flexible, however when combined with the CIA hardware timers it can be made to work very reliable.

To increase data transfer efficiency, instead of using one signal for a 1 and one signal for a 0, we choose a sightly more efficient method of assigning 4 signal frequencies to a set of 2 bits.

Some might still ask "Why not use the commodore tape protocol?" Well, simply because it is way too complicated for this purpose. It's way too slow and way to redundant. It would be impractically slow and let's be honest, if in an entire image a few pixels contain errors does it really matter, chances are that nobody will ever notice. A slow system, is annoying all the time.

So, you might wonder… why in 2022 would someone want to do this kind of project, well… because it is fun to make it work anyway. "Doing the impossible..." is a bit of an overstatement, but doing the unexpected is exactly the purpose of this project, doing something nobody thought the C64 could do or hasn't been done yet, there lies the challenge and therefore all the fun.

10 Tape signals

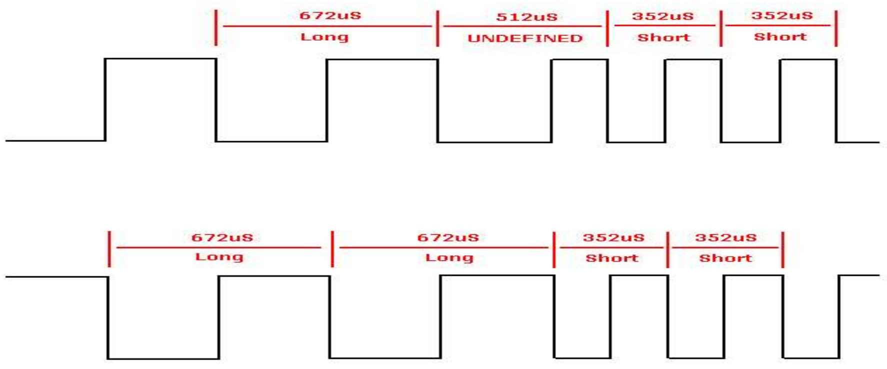
10.1 Polarity of the signal

Now you might wonder why the polarity of an AC signal stored on tape is important, after all we are storing purely symmetrical signals. Well, the answer to that question lies within the way the C64 is able to read those signals. The C64 can NOT detect the level of the signal on a tape, it can only see it’s falling edge.

To add to the confusion of signal polarity, the C64 outputs a write signal to the tape that is the inverse of the signal read back from the tape. Not a big deal, but if you do not know this then this is a pitfall you can easily step into. Which is exactly what I did during one of my other tape related projects. I simply did not understand why my signal could not be read back into the C64, I was mimicking exactly what the computer was sending. The reason was simple in the end. I needed to invert it.

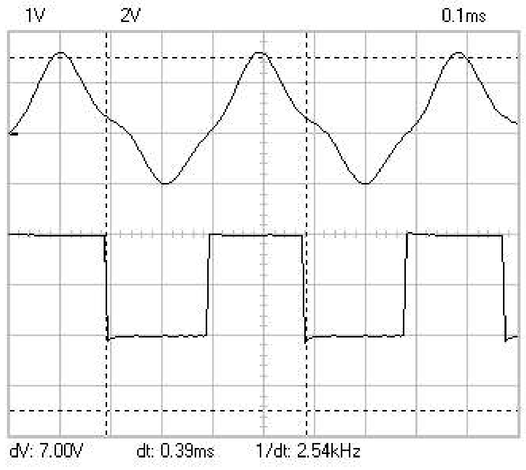
Now it does not matter if you create signals digitally or using an analog tapedeck with an amplifier that inverts the signal, the problem is simple. Wrong polarity = non working system.

When you consider that **the C64 only can detect the falling edge of the read tape signal** (digitized tape output is connected to the CIA's FLAG input) it becomes clear that the C64 measures the time of a complete (digitized) signal. It does not detect the individual high or low levels of the pulse. It detects the falling edge of the pulse, which is very smart as it is the absolute minimum you’d require. But, because the falling edges are triggered (and not the width of the high or low pulse) it is important that the signal is presented in the proper way. In other words in the correct polarity. When (for whatever reason) it is inverted then the C64 can no longer measure the duration of the signal correctly as it will be combining two signals into one. Below are 2 signals. As you can see in the picture below (Commodore C64 tape format), the top signal should be inverted, the bottom is correct. When measurements of the signal width cannot be measured correctly it will result in undefined timing values causing the C64 to generate a LOAD ERROR.



10.2 Commodore tape signal encoding

This method is not used by the StripStream protocol, but to get an idea of how the hardware works under normal conditions is important to understand the significant improvement the StripStream protocol provides.



Shown here is the pilot wave (a.k.a. trailing tone). It is the long beep at the start of each program. As you can see, the sinewave it not a very clean one. It looks more like a triangle. Fortunately this is not a real problem as the signal is digitized by a Schmitt trigger circuit into a digital high or low value.

The upper signal is the signal as read from tape. The lower signal is the digitized value that is send to the computer. The signal shown is the signal defined as a series of S's. The S stands for Short pulse (more about this on the bottom of this page).

The Commodore system uses three different frequencies that and are always used in pairs. The only exception is the pilot tone because this is just a series of S's for several seconds. The pilot tone is used for the C64 to lock into the tape signal. This because the tape could have been recorded on a different recorder with a different speed. Since the Pilot tone is of a defined frequency, the C64 can measure its frequency and compensate for the difference in tape speed. This is why the pilot tone is heard for several seconds, but also to allow the tape to stabilize (the motor might have been just started and might be slowly speeding up) and achieve a proper and stable speed.

Pulse duration specifications

Theoretical values for S, M and L as described in a “Data Becker” book (C64, VIC20, PET, C128)

Short	: a sine of 2840Hz (=>352uS), this means that a high-pulse takes 176uS and that the low -pulse takes 176uS
Medium	: a sine of 1953Hz (=>512uS), this means that a high-pulse takes 256uS and that the low -pulse takes 256uS
Long	: a sine of 1488Hz (=>672uS), this means that a high-pulse takes 336uS and that the low -pulse takes 336uS

The following combinations are defined:

Pilot wave		: S
Byte marker waves	(more data follows)	: LM
Byte marker waves	(end-of-data)	: LS
Data waves	bit = 0	: SM
Data waves	bit = 1	: MS

10.3 StripStream tape signals

This chapter describes the origin of the definitions of the various pulse width values used.

Because we must send some kind of pulses to identify a logic 0 or a 1, we simply can send two different frequencies, a high frequency=1 and a low frequency is a 0. If the max. allowable frequency on the tape is to be 5kHz, because of technical limitations of the tape, recorder and the C64 (which has it's raster interrupts and bad lines). Because the tape can vary in speed for various reasons we must allow for a very large margin of error, so let's assume we must be able to cope with a speed error of +/- 10%. Which is huge, but we want this to work on real hardware so we can't be safe enough.

A single bit signaling setup might look like this, it is a concept used by many fastloaders that have the screen disabled (so no bad-lines and no raster interrupts). But, in the project we have the screen enabled and we DO use raster interrupts. Let's assume a raster interrupt that takes 60 cycles, 60 us for simplicity. So, as we can see, the additional 60us causes a problem as the max. time of the “1” overlaps with the min. time of the “0”. Completely unusable!!!!

Theory			**Practical**		
Value	**.Tap**	**Freq/Period**	**Minimal time** (period-10%)-60us	**Maximum time** (period+10%)+60us	**Timer checkvalue**
1	24	5132Hz / 195us	120us	274us	???
0	39	3240Hz / 309us	218us	339us	???

```
Average datarate
max. datarate     = 5132 * 1 bits/sec = 5132 bits/sec       =  641 bytes/sec
min. datarate     = 3240 * 1 bits/sec = 3240 bits/sec       =  405 bytes/sec
                                          --------------------
                                 average = 615+405 / 2     = 523 bytes/sec
```

Now although (despite from being unusable in this project) the transfer itself would be mighty fast if we could use it. but, we just can't. regarding the speed, we must also keep in mind that this is without the overhead of byte separation (like a byte start/stop indication). But that’s all just an extra layer which we can ignore when figuring out which signaling scheme is the best for a simple tape streamer project like StripStream.

Because the above setup cannot handle the raster interrupt (and/or bad-lines). If they would occur they would add additional counted cycles to the end of a pulse, but also subtract (or to be more precise "missed") from the pulse directly after. Because counters keep counting and the tape keeps on running. A bad line takes 43 cycles max (since we have no sprites, otherwise it would be even more). Fortunately we can prevent that a bad line and raster interrupt occur directly after each other.

Now, let's assume a dual bit signaling setup and do take the wasted interrupt cycles into account (we can ignore the bad-lines, simply because the bad-lines are taking up less then 60us but also because the raster interrupt and bad-lines will never coincide, simply by choosing the location of the raster interrupt at a location exactly in between two bad-lines.

Calculations below are for a PAL system. Again we'd allow for a tape speed variance of 10%. Should you experience a 20% tape speed error then repair your datasette and use a decent first generation copy of the tape.

Formulas used: TAP file time= value*8/985248
minimum time = (value*8/985248)*0.9 - 60
maximum time = (value*8/985248)*1.1 + 60
timer check value = maximum_time/8*985248

Theory				**Practical**	
Value	**.Tap**	**Freq/Period**	**Minimal time** (period-10%)-60us	**Maximum time** (period+10%)+60us	**Timer checkvalue**
00	24 = $18	5132Hz / 195us	120us	274us	<274us = 33 = $21
11	46 = $2E	2677Hz / 374us	276us	470us	<470us = 57 = $39
10	73 = $49	1687Hz / 593us	473us	712us	<712us = 87 = $57
01	106 = $6A	1162Hz / 861us	714us	1007us	<1007us = 124 = $7C
- -	- - -	- - - - -	- - -	- - -	- - - - -
SYNC	147 = $93	838Hz / 1299us	1015us	- - -	>1015us = 125 = $7D
			- - -	1373us	<1373us = 169 = $A9

Actually we have 5 different pulse types, of which the slowest is used for syncing purposes, because we need a pulse that is not to be confused with data. And because the sync-pulses are less used than the data-pulses, it doesn't matter that much if the sync-pulse is relatively slow.

Now although our lowest frequency is much much lower then in the previous example, it doesn't need to be problematic. If the data isn't all ones or zeroes, but nicely random, this would result in an average data rate that is even higher then with a simple two frequency system.

The definition of frequency per bitpair is determined based upon the example comic I made, I analyzed the data and came to the conclusion that the usage of bitpars is as distributed as follows 00=40%, 01=14%, 10=16%, 11=28%. So knowing that a file consist mainly of bitpair 00 it makes sense to give that bitpair the shortest possible signal. And honestly it makes a lot of sense that 00 is the most used, considering that all packets start with 00 and lot's of text holds 00. But also imagewise, there are lot's of big areas that are completely black and lot's of areas that are completely white. The large black separator bars are already optimized so they don't count. So with this knowledge it is wisely to couple 00 and 11 to the fastest signal (a.k.a. highest frequencies or shortest pulses).

Average data rate
max. datarate = 5132 * 2 bits/sec = 10264 bits/sec = 1283 bytes/sec
min. datarate = 1162 * 2 bits/sec = 2324 bits/sec = 290 bytes/sec

Average datarate = (1283+290)/2 = 786 bytes/sec

Keep in mind that one pulse identifies 2-bits. So actually one pulse moves twice the amount of data. Which results in a much larger average throughput. Now you might wonder what would happen if we double that again, however that would double also the number of required frequencies and all with a safe (approx. 20%) separation, it would make the lowest frequency almost impractically slow, somewhere in the region of 250Hz.

max. datarate = 4926 * 3 bits/sec =14778 bits/sec = 1847 bytes/sec
min. datarate = 250 * 3 bits/sec = 750 bits/sec = 93 bytes/sec

Average data rate = 1874+93 / 2 = 983 bytes/sec

Now that does look like a significant improvement! BUT... handling data in groups of 3 bytes will result in some very long and possibly messy code, code that might take too long to execute. For the situation of the fastest possible pulse of 5kHz. 200us are easily spend on a C64 with raster interrupts and bad lines. And although a series of ones would seem amazingly fast... (keep in mind that we are working with a C64 with an enabled and rastersplit screen here) a series of zeros would be unacceptably slow. A situation not unimaginable if we send images that are completely black or white. So to keep things simple, practical and more importantly "fun", the dual bit signaling setup will be perfect for our needs and therefore will be used.

Synchronization signal
Regarding a signal for data synchronization, there is a bit of a problem. We are sending our data in the form of vertical pixel lines. And after 8 lines, the screen needs to be shifted. We could send a very long (low frequency) pulse to act as a sync. so that during the time of copying the screen data, no data is send by the tape.

Meaning that the sync pulse needs to allow for the copy action to take place. But copying of 320*98pixels, or to be more precise 312x98 (3822 bytes) takes quite some time, at least 8 cycles per byte (slightly more due to some overhead in the partially unrolled copy loop), meaning that the copy action to perform the scrolling requires approx. 32ms. Which would require a sync pulse or delay of about 32ms = 31Hz, way below the frequency range of the datasette.
However, copying so much data poses another problem, because moving data on the screen would result in artifacts, simply because the VIC-chip draws the screen faster than that we can copy it. So double buffering is a necessity anyway. Change one screen, while showing the other. This also means that we do not need to copy the screen in one quick burst. We can simply copy parts of the screen in advance. Meaning that we do not need to copy 3822 bytes once every 8 lines, but only need to copy 3822/8, approx. 480 bytes on every line. Spreading the load significantly. Meaning that a sync pulse of 32/8ms will be more than enough to be detected and do the required copying. A pulse of 4ms has a frequency 250Hz, which is in the allowable frequency range of the 1530. Or at least you would expect, because this doesn’t work either. Apparently even this is still to low to generate a reliable pulse.

So we need to do something else, something that DOES work with all datasettes and is not critical in it's timing regarding the CPU activities of the StripStream reader program. So we choose a pulse that would be the fifth pulse in the row of 4 pulses separated equally (see table). But because that pulse would be way too fast, we need to send more than one, in fact we send a whole bunch of them, followed by a packed ID. The C64 is supposed to handle this as follows:

- detect at least 3 sync pulses in a row before allowing the acceptance of any other kind of pulse.
- if after a sync pulse a pulse other than a sync pulse is detected, the pulse should be treated as if it was the first pulse of the 8-bit packet-ID.

11 Packet definitions

This StripStream protocol sends its data in the form of “packets”. These are separate fields of information that can be processed independently. The great benefit of packets is that they can be ignored if not recognized. Meaning that it should be possible to extend the format with new package types if desired, without breaking backward compatibility.

Because a StripStream book is readable from any point in the file (there is no need to start reading from the beginning, so you can stop the tape and continue where you stopped a few days later) the packages defined below are to be send on a regular basis. Preferably once every frame.

Because data storage on tape isn’t always 100% error free, it is to be expected that loading errors will occur. This does not have to be a problem, a black pixel in the location of a white might not even be noticed. And because most of the packages are to be send multiple times, if an error is displayed, it will be overwritten with new data the next frame, so it disappears. By leaving out error checking things are relatively simple, which is important for a 1MHz computer, which needs to process it all while also loading the data and updating the display at the same time.

Each packet consists of 3 important sections:

- The packet-sync is a low frequency signal of … us wide.
- The packet-ID is byte which value indicates the packet-ID
- The packet content is the actual payload, it may vary in length depending on the packet, but the size of the packet content is exactly defined by the packet definition of that specific packet-ID.

So when analyzing the content of a StripStream tape, the layout of the data on the tape might look like something like this.

Packet sync	Packet-ID	Packet content

Packet sync	Packet-ID	Packet content

Packet sync	Packet-ID	Packet content

Packet sync	Packet-ID	Packet content

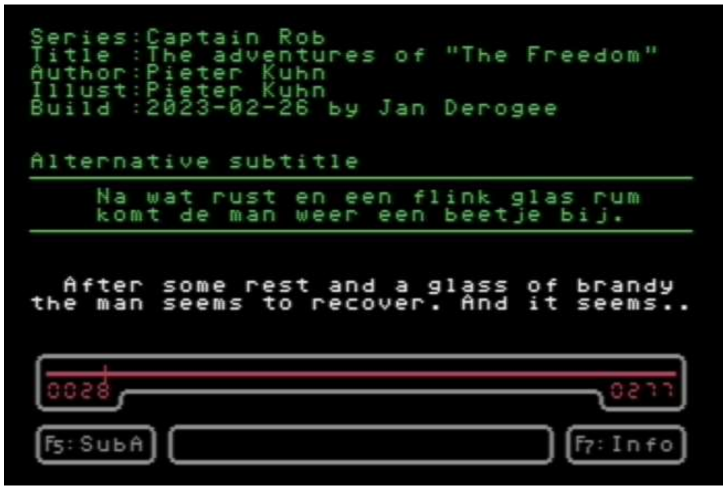
Packet sync	Packet-ID	Packet content

Packet sync	Packet-ID	Packet content

Packet sync	Packet-ID	Packet content

11.1 Meta data

In order to identify a production, the credits packet might be of help. By holding the original title (as long as this fits into the field) and the name of the author. The production date is important as it allows for identification of older and newer (error corrected) productions. When the user presses F7 on the Commodore 64 during playback of the StripStream file the HIRES ection of the screen will change into a green text section and shows the loaded meta data.



```
Series:Captain Rob
Title :The adventures of "The Freedom"
Author:Pieter Kuhn
Illust:Pieter Kuhn
Build :2023-02-26 by Jan Derogee

Alternative subtitle

     Na wat rust en een flink glas rum
     komt de man weer een beetje bij.


   After some rest and a glass of brandy
the man seems to recover. And it seems..
```

Although this information is important, it isn’t important enough to send it after every frame. Therefore it is best to “interleave” it in between the frames. Sending one item of the meta data after each frame instead of sending all the meta data after every frame. So for example:

frame n, meta data “series”, frame n+1, meta data “title”, frame n+2 meta data “Author” etc. This will reduce the amount of transferred data significantly and. If the user views the meta data info screen directly after starting the tape, they might notice that the screen requires some time to fill up the screen with all the possible information. To save tape space, it only loads one meta-data item every frame. As soon as all the items are loaded this isn’t even noticeable any more as the meta data will not change during the entire project.

Also shown underneath the meta data, is the alternative subtitle, this way the user is able to view both subtitles at the same time.

The meta data consists of various packages, one package type for each possible item, below a description of the packages in the same order as they are shown on the screen.

Series **(packet-ID = 0x10)**		
Byte	**Content**	**Information**
0..32	Series	The series is exactly 33 bytes of data that is directly printed to the screen locations that are intended for series display.

Title **(packet-ID = 0x12)**		
Byte	**Content**	**Information**
0..32	Title of the book	The title is exactly 33 bytes of data that is directly printed to the screen locations that are intended for title display.

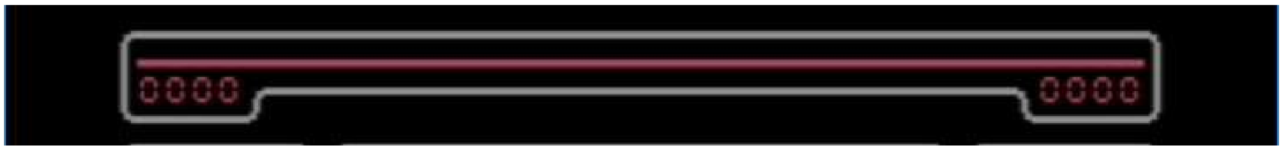
Author **(packet-ID = 0x14)**		
Byte	**Content**	**Information**
0..32	Author	The author is exactly 33 bytes of data that is directly printed to the screen locations that are intended for author display.

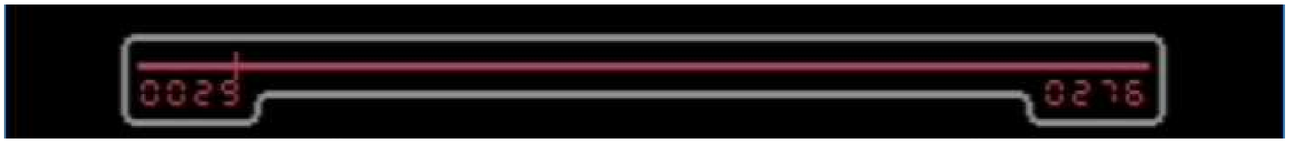
Illustrator **(packet-ID = 0x16)**		
Byte	**Content**	**Information**
0..32	Illustrator	The illustrator is exactly 33 bytes of data that is directly printed to the screen locations that are intended for illustrator display.

Build **(packet-ID = 0x18)**		
Byte	**Content**	**Information**
0..9	Build date	The date is exactly 10 bytes of data that is directly printed to the screen. Although it can hold any kind of character (like text) it is to be used for display of the build (save) date of .TAP file. The suggested format is: YYYY-MM-DD
10..18	Build by	The build by is exactly 19 bytes of data that is directly printed to the screen in the appropriate location.

11.2 progress-bar

When reading a real paper book or magazine is easy to get a feeling of how much there is to read. So you can easily decide to stop or continue reading is you are almost finished with the story. For a digital format, there is no clear indication, unless a progress-bar is used. But a progress-bar is not enough. For a very large production, a progress-bar simply cannot indicate the exact location in the file, it would be to crude. With only 38x8 possible pixels to place the indicator. Therefore there are also frame counters. Like the current frame and the number of remaining frames.



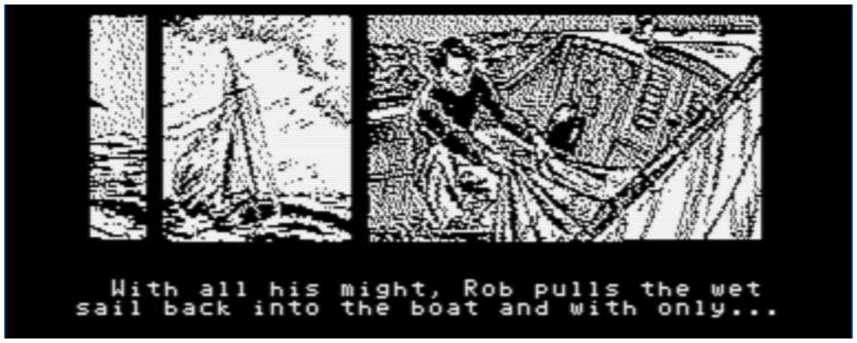


Above are shown two examples of the progress-bar as it looks like in the StripStream player. When no info has been loaded yet, the progress bar does not show the vertical indicator line and the values are all "0000". On the left is the current image counter, on the right is the remaining images counter. This all gives the user a pretty good idea to the user of it's position within the StripStream file. Below the packet that defines the progress bar.

progress-bar **(packet-ID = 0x20)**		
Byte	**Content**	**Information**
0..3	Current frame	This is a four digit number (value send as directly printable text and not as a value) that indicates the current frame.
4..7	Remaining frame	This is a four digit number (value send as directly printable text and not as a value) that indicates the number of remaining frames.
8	Pointer position	The position of the pointer on the bar of 38 possible positions
9	Pointer character	The character to be printed at the pointer position, since a character is 8 pixels wide, this allows the resolution of the pointer to be multiplied by 8, since there are 8 possible pointer characters to be used, each one pixel shifted.

11.3 Text for subtitle field

Subtitles are a way of creating readable text. The problem with the limited resolution of the StripStream images is simply that large text balloons would require too much space, leaving no room for the artwork to be displayed. Therefore there are no text balloons to be used in StripStream productions, the text is simply displayed underneath the artwork in the text-region of the display.


With all his might, Rob pulls the wet
sail back into the boat and with only...

The text in subtitles is to be aligned (padded with spaces) so that all text is centered. This way the player software does not need to worry about alignment and it can simply print the text as found in the packet. The padding is automatically done by the StripStream editor software, so the creator of the StripStream file itself does not need to worry about that.

There is support for two different languages, language-A (also referred to as the primary language). By supporting multiple languages a release can be used by a wider audience. For instance subtitle A could be English and subtitle B could be Dutch. When multiple language support isn't required then do not send the packages for the secondary subtitles. Although the best solution would be to frequently send an empty* subtitle package as this would erase the fields from previous content.

Subtitles A (primary language) **(packet-ID = 0x24)**		
Byte	**Content**	**Information**
0..39	Subtitle line-1	Text that relates to the story line and corresponds with the images *
40..79	Subtitle line-2	Text that relates to the story line and corresponds with the images

Subtitles B (secundary language) **(packet-ID = 0x26)**		
Byte	**Content**	**Information**
0..39	Subtitle line-1	Text that relates to the story line and corresponds with the images *
40..79	Subtitle line-2	Text that relates to the story line and corresponds with the images

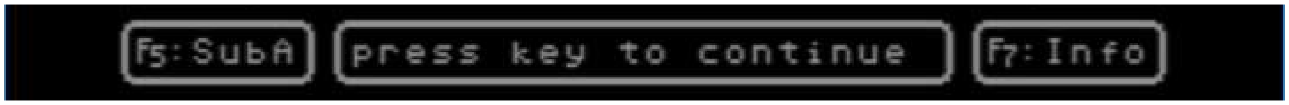
*: If no subtitles are required, there is no need to send all 80 bytes, when the first byte of the subtitle field is the value 255 (FF hexadecimal) both lines of the subtitle field will be cleared.

Note:

The character set used only consists of slightly less than 96 characters. So technically it would be possible to use a 7 bit encoding system to transfer the subtitle data. Unfortunately, since the data is always send in groups of two bits there is no advantage in using 7 bits instead of 8 bits. Because we need to send 4 consecutive 2-bits signals anyway in order to send the 7-bits code. In other words, we cannot reduce the data of the subtitles by discarding a bit. Although technically a character set under 64 characters would be possible (which would allow for a 6 bit code to be send instead of an 8 bit code) this also means that the text would always be in upper or lower case. For readability a mix of upper and lower case characters is preferred and therefore a 6-bit charset can not be chosen.

11.4 Instruction field text

This is text that instruct the user on what to do next. There is a special field reserved for it in the played, it is in the center of the bottom of the TEXT-based screen. Below a screenshot of that section of the screen.



The text inside the instruction field can be anything. This allows a StripStream release to be better suited for a non English release. Although all the other fields, which have fixed text, are still in English. Anyway, allowing any kind of text to be send to this field make the program just a little bit more flexible, doesn't it?

Instruction field text (packet-ID = 0x28)		
Byte	**Content**	**Information**
0..21	Instruction text	Text describing the user what to do next, for example: "press a key to continue" or "hold key to pause"

11.5 Instruction field clear

This packet clears the instruction text field.

Instruction field clear (packet-ID = 0x2A)		
Byte	**Content**	**Information**
n.a.	n.a.	This package is empty as it contains no data bytes

11.6 Request player to continue

This package makes it possible for the player to be paused, but only when this package is detected. If this packet is received the player stops the motor of the datasette and keeps looping until the user presses a key on the keyboard. When a key is pressed the datasette motor is started again and the player continues.

This package is intended to be send AFTER the sending of a complete image. Although it can be send more frequently if desired, that would only reduce system performance.

Note: if a datasette is allowed to be stopped, it will need time to get back up to speed again and it might not be able to read a few of the following packets correctly from tape. Therefore make sure to add additional sacrificial or "non-critical" packets before sending "critical data" (like images). See the packet "separator line" for more details.

Attention: this package also clears the instruction text field once a key press has been detected. Therefore there is no need to send the instruction field clear package.

Request to continue **(packet-ID = 0x2E)**		
Byte	**Content**	**Information**
n.a.	n.a.	This package is empty as it contains no data bytes

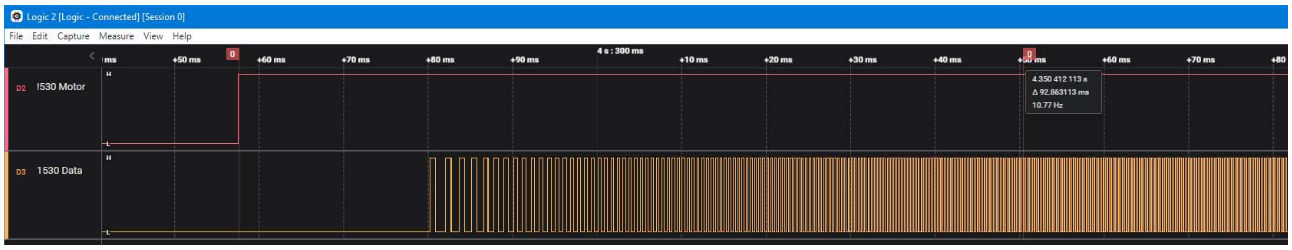
Attention:

Keep in mind that after this package additional tape space is to be reserved in order to allow the tape to stop (because the tape doesn't stop immediately because of the inertia of the entire mechanism) and to allow the tape to start (when the motor starts the tape isn't immediately playing back at the correct speed, also because of the inertia). The second of wasted tape space is to be filled with a non-data signal in order to prevents data loss.

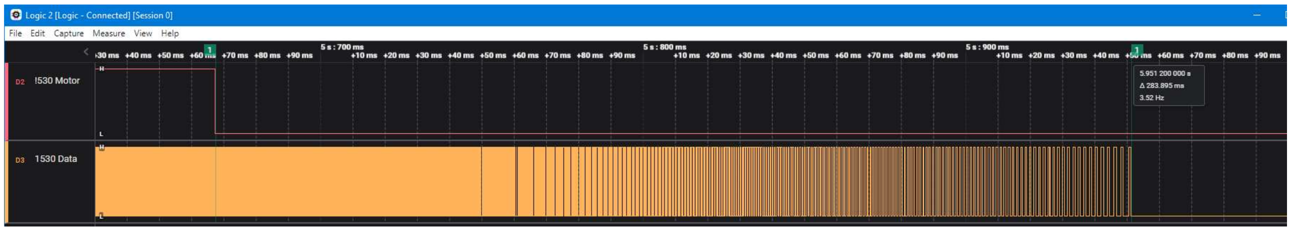
If there is no safe space reserved on the tape for locations where the tape is expected to be stopped and started then data will be lost, which would be unacceptable. Adding a safespace solves this problem and is simple yet effective.

The tape cannot stop instantly and it isn’t instantly back to the perfect speed when the device is ordered to stop or start. Below are two measurement on a 1530 datasette and a 3150Hz reference tape.

As can be seen below, there is a delay between the starting the datasette motor and the moment of available AND stable data. As soon as the motor starts to spin data comes from the device, but it takes a little longer before the device is actually back up to speed. The measurement below indicates that it requires at least 100ms. Which, is pretty fast for this kind of device.


Logic 2 [Logic - Connected] [Session 0]
File Edit Capture Measure View Help
4 s : 300 ms
D2 !530 Motor
D3 1530 Data
4.350 412 113 s
Δ 92.863113 ms
10.77 Hz

Stopping the tape takes considerable longer. This is because the device is not actively driven to the new state as it would by starting the tape. By stopping the tape, the power is cut, the motor is no regulated any more and runs under the power of the flywheel and the inertia of the motor itself. This stopping action seems to require at least 300ms.



So to be on the safe side, for the device used here, we need to reserve a safe space on the tape where no data is present, so that the tape could be stopped and started at that location safely. Simply speaking, that would be a safespace of 100ms+300ms = 400ms. But that isn’t entirely true. Because when the tape runs slower, it consumes less tape, meaning that time is stretched. So effectively the safespace on the tape could be a littlebit shorter. But how much? If we count the cycles since the tape is stopped until the data stops, we count 430 cycles. The tape is a reference tape with a very accurate signal of 3150 Hz which we can use to measure the length of the tape (measured in ms of recording time). The 430 cycles of a 3150Hz signal required 137ms to be recorded. So in reality a total safespace of 250ms should be enough for this device.

HOWEVER… this is just one measurement and just one device, there are a lot of 1530 or similar devices on the planet that might respond slightly different. To be on the safe side, we should keep a bigger margin for the safespace. Otherwise problems will arise in the field.

But keep in mind that a safespace of 1 second, in a release with 300 images, consumes 300*1sec=5 minutes of tape. That’s 5 minutes of tape wasted only to allow the tape to stop and start. This is perhaps a bit too much, therefore the StripStream editor software uses a safespace of 500ms, this is twice the optimal safespace for a perfectly functioning 1530 datasette.

11.7 Separator line (a.k.a. vertical black line)

The image scrolls from the right of the screen to the left of the screen by the insertion of a black line.

<u>Note:</u> this package is to be considered non-critical. Meaning that if this packet is missed or lost there is no direct problem and the user will most likely never even notice if there are 16 black lines shown or 14. This is important to know for situations where the datasette is to be allowed to be stopped. Once the datasette is started again it needs to get up to speed and if there is data on the tape during that time, the loading of it might result in corrupted packets that cannot be processed. The exact amount of time (or length of tape) the datasette requires to achieve the correct and stable speed, depends on the condition and mechanism of the datasette used. Therefore always make sure to send at least 8 of these separator packets before sending any important data (like a new image). Because if one or two of these kind of packages re missing, nobody will ever notice.

However, to prevent confusion, it is best if that situation never occurs and that the tape does not contain any data around the area it might be stopped/started. A 1 second space on the tape should be reserved at the places where a tape-stop/start is to be expected, for example, directly after an "request to continue" command.

Black separator line **(packet-ID = 0x30)**		
Byte	**Content**	**Information**
n.a.	n.a.	This package is empty as it contains no data bytes

11.8 Image line (a.k.a. vertical pixel data)

The image scrolls from the right of the screen to the left of the screen. The data is scanned vertically. Meaning that the image data consists of a stream of vertical lines. Where each line is a packet.

The reason why an image is send as a series of vertical lines packets and not as one big packet holding the complete image is simply because of safety reasons and to allow to send other kind of packages in the middle of the image. For instance subtitles are to be send somewhere between the start and the end of the image (preferably the middle). This provides a more convenient way of reading. The safety aspect is directly related to the reliability of tape storage without error correction. If the image was to be send as one big set of data AND one pixel is missed or added (due to tape dropout) then the whole image would shift/skew one pixel, making the final image look very weird. By sending the data in packets of vertical lines, a single error will never propagate through the entire image and will just look like a funny pixel.

In case of emulators, this would not be a problem at all, but this protocol was designed to be used with real hardware in a real world with real problems.

Image line **(packet-ID = 0x32)**		
Byte	**Content**	**Information**
0..11	Vertical pixel data	A line consists of 12 bytes, of one bit per pixel. Meaning that the vertical resolution is 12*8 = 96 pixels. Bit = 1, results in a white pixel Bit = 0, results in a black pixel

11.9 Audio functionality

Perhaps some day in the future, audio support might become available. However, the datasette isn’t an audio capable device, it simply was never designed for that purpose. But it could be a thing that might be fun to experiment with in the future. Therefore the definition of an audio packet has already been reserved.

Audio (packet-ID = 0x0E)		
Byte	**Content**	**Information**
n.a.	n.a.	This package is reserved for future use

12 Speech synthesizer support

If a “Serial Speech Synthesizer SAM” or a “Votrax type 'n talk“ module is connected to the userport of the Commodore 64 then the subtitles can be heard as spoken text via that module. This because as soon as a frame is loaded and the message “press key to continue” appears the tape is stopped, this allows for the C64 to send the shown subtitles as ASCII data to the userport in the form of RS-232 serial data. This take some time (the serial routines are just simple bit-banging code and therefore time consuming). This means that no data from the tape can be received during the transmission of the serial data to the userport. However, it really doesn’t take that long from a human perspective, since 81 characters (40 text characters + 1 carriage return) at 2400bits/sec. require approx 350msec. Which is why there is no real need to disable this functionality for the users who do not have such hardware connected to their userport. If they don’t have the module they do not hear anything and the additional delay between the “continue” key press and continuation of the tape playback is almost unnoticeable. For those who do have such a module connected to the userport of their C64 but don’t want to listen to it, they can always turn down the volume of the monitor.

13 Interrupts

The StripStream player has a split screen, so that we can display HIRES image and ordinary text onto the same screen. This means that we need to split the screen using interrupts. BUT… interrupts take time, and “bad lines” also take time (since they disable the CPU for a period of 40-43 cycles). And what we certainly do not want to happen is that bad lines and interrupts coincide or happen very shortly after each other, since that would cause a delay larger then the 60us which we have allowed in this protocol.

Bad lines do not happen when the raster is in the border area, so there are no bad lines when the raster value is smaller than $30 and larger than $F7.

So if we want an interrupt to happen it would be best to make it happen just in between two bad lines. Bad lines are very predictable, so we can do this. Bad lines happen when the lowest 3 bits of the raster counter are equal to the lowest 3 bits of $D011 (the lowest 3 bits of $D011 are the scroll Y register).

Now common sense would make you believe that the scroll register is 0 when you switch on the computer, since there is no scrolling, so it should be at some start position. However, this is not the case, since the default scroll position is in the middle of the scroll-able range. By default the scroll Y register is 011 or 3. Meaning that when the raster counter hits a line that ends with $X3 or $XB we have a problem, since it will be a bad line.

So if we make sure that our interrupts occur at a line ending at $X7 or $XF we should be at the most ideal moment in time (the furthest away from a bad line).